Interaction in Distributed Virtual Environments

Mashhuda Glencross, Miguel Otaduy and Alan Chalmers

Contents

Abstract

This course will describe the main challenges faced when building engaging shared virtual environments supporting complex behaviour and interaction, and provide discussions on techniques that can be adopted to support some of these. In order to build such environments, it is necessary to combine high quality graphics, better modes of interaction, rich behavioural simulations and appropriate distribution strategies.

After introducing the field of interaction and rich behaviour in collaborative virtual environments, we cover the main issues in three parts. First we look at techniques for improving the user's experience by using high-fidelity graphical rendering, and explore how this may be achieved in real-time through exploitation of features of the human visual perception system. We examine also how additional sensory modalities such as audio and haptic rendering may further improve this experience. Second we consider issues of distribution with an emphasis on avoiding potential pitfalls when distributing complex simulations together with an analysis of real network conditions, and the implications of these for distribution architectures that provide for shared haptic interaction. Finally we present the current state of the art of haptic interaction techniques. In particular the motivations for perceptually-inspired force models for haptic texture rendering, interaction between such models and GPU techniques for fast haptic texture rendering.

The objective of this course is to give an introduction to the issues to consider when building highly engaging interactive shared virtual environments, and techniques to mediate complex haptic interaction with sophisticated 3D environments.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Three-Dimensional Graphics and Realism]: Virtual reality

1. Introduction

The real-time performance requirement of interactive virtual environments has resulted in the rendering requirements generally dominating computational costs. Thus many virtual environment demonstrator applications constitute walk-throughs involving limited direct interaction with the environment. The real world however is complex, and contains many stimuli which simultaneously engage all our senses; sight, sound, touch, smell and taste. These combine with our experience of the world to give us an intuitive understanding of realism. Immersive virtual environments seek to provide a sense of being physically within a synthetically generated space, through a combination of multi-sensory stimuli, and isolating the user via the use of a head-mounted display. Unfortunately virtual reality technology is still cumbersome, the quality of head-mounted displays is still relatively poor, and the technology is still costly.

Studies of presence [WS98, Sla99] including those which borrow from gaming theory, have shown that user's need not be outfitted with an immersive setup to become engaged in an environment [DH00, Man01a, BC04]. The level of engagement participants exhibit during game play, often using desktop computers or play stations, may be considered to be indicative of a sense of presence. A key factor contributing to this, is that during game play, users actively participate to carry out a particular task (usually shoot the bad guy). Typical first person shooter games have a fast pace with the user having to consider other characters shooting at them, explosions, and sound effects while trying to achieve their objective. There is a lot of activity to occupy players leading to a high cognitive load. In a gaming context however a large number of assumptions based on the storyline can be made about the actions the player is most likely to perform, and therefore simplified simulation of the environment can be employed.

On the other hand since it is impossible to envisage all the demands of every conceivable application, in the context of systems to build a wide variety of virtual environments, few such assumptions are possible. Consequently, it is not so straight forward to build such compelling and engaging environments. This leads to the question of how to make serious virtual reality applications more engaging and intuitive to use? The answer is to improve the correspondence of the virtual environment to the real world, through a combination of sophisticated interfaces for advanced human computer interaction coupled with good rendering and behavioural fidelity. In the following sections we describe what we mean by these.

2. Complex Interaction in Shared Virtual Environments

Despite a wide range of three-dimensional (3D) input devices being available (see Hand [Han97] for a good survey), the most common methods of interacting with a 3D environment is still via a standard mouse or a keyboard. Typically participants press keys and mouse buttons to invoke modal changes and use a standard mouse to provide positional information during drag and drop or navigation tasks. A possible reason for this may be the cost of more sophisticated input devices, their ease of use from a developers point of view, or perhaps even user familiarity.

Due to a more intuitive spatial mapping between the real and virtual world 6 degree of freedom (DoF) input devices, such as a SpaceMouse or electromagnetically tracked devices, provide a more intuitive mode of interaction in 3D environments. However, participants often exhibit difficulties with depth perception in virtual environments [RGA95]. A specific performance evaluation of input devices typically used in 3D applications was carried out by Roessler [RG98], and showed a paradoxical difference (which could be attributed to poor depth perception) between the actual and perceived accuracy of a tracked button used by test subjects. By engaging another sensory modality to support visual cues 3DoF and 6DoF force feedback (or haptic) devices could improve this situation, and in doing so enable better perception of the spatial layout of entities in 3D environments.

Complex interaction is not purely related to input and output devices, but also potentially has a causal relationship to the state of entities with which the user interacts in the environment. This state change may be reported back to the user through multi-sensory feedback such as colour changes, and the use of auditory or touch (haptic) feedback. In a large number of virtual environments, participants may change simple attributes of entities. For example by selecting and moving a virtual representation of a baseball, its positional attributes will change. More complex entities which have physical characteristics allowing them to deform or flow, may not only have their position altered by user intervention but also their structure. Consequently, this in turn impacts upon the values provided to the algorithms used to compute the structure of the entity.

Another possible source of complex interaction in virtual environments can be through communication with other remote participants. These may range from simply saying ‘hello’, to performing a complex sequence of tasks to achieve a shared goal. Few virtual environments employ many of these different types of complex interactions in conjunction with each other. Engaging environments should however ideally enable such interactions in behaviourally-rich environments, which may potentially contain a variety of dynamic entities whose state and/or structure may be changed. It is therefore useful to consider how rich behaviour may be represented in frameworks for building virtual environments.

3. Behaviourally-Rich Shared Virtual Environments

Behaviour-rich virtual environments ideally model the Newtonian physics of the real world. However it is far beyond computational capabilities to faithfully simulate all the forces and torques acting on bodies, and correctly compute motion and deformations, for every entity in a complex virtual environment therefore ideally simulations need to be simplified [Gle00]. Provided such simplifications can still be used to compute plausible or perceptually correct behaviour, it may be possible to dynamically simplify the underlying structure of the models according to run-time requirements [GHP01]. However, due to the performance demands of simulation and difficulty of dynamic complexity management, scripted behaviours [CPGB94] are often invoked upon specific events or actions, for example a user picks up a specific entity which (when selected) records all the user’s activities until it is dropped (or unselected) [GFPB02]. This type of event based invocation of behaviour is a good way to represent a wide range of common background activities, such as a vehicle driving along a road or the bounce of a ball. Alternative approaches exploiting features are more intelligent, and determine the behavioural response of an entity from semantic information in the model. For example, a desk drawer can be opened and closed because it is a desk draw [KT99].

In prototyping or training applications however, if participants actions have an effect on the motion or structure of entities, scripted behaviour in itself is not sufficient. Feature based and constraint methods have been employed to perform rigid body assembly tasks however these methods require enough semantic information to be present in the models, and cannot simulate the deformation characteristics of complex flexible components [MMF03]. The behavioural response in such applications may be specified through a number of rules and physically based simulation models. A variety of software frameworks exist to simplify the development of behaviourally-rich collaborative virtual environments [GB95, Hag96, PCMW00]. Many of these incorporate a sophisticated world model combined with spatial subdivision schemes, to limit the scope of particular characteristics. For readers wishing to know more about these, Pettifer presents a comprehensive survey [Pet99].

Pettifer et al.’s [PCMW00] Deva system developed at the Advanced Interfaces Group at Manchester adopts essentially a *client-server* distribution architecture (detailed later in these notes), but with a flexible configuration for communicating state changes of individual entities in a shared environment [Pet99]. The system achieves this by decoupling behaviour into an objective (semantic state) and subjective (perceptual state) component. The single objective part of entities resides on a *server* which maintains consistent state, and in order to either render or interact with the entities each participant’s *client* creates corresponding subjective parts. These possess behaviour and can act plausibly until otherwise instructed by the server, and so their state may subtly and briefly differ from the objective reality as shown in Figure 1. It is the responsibility of the objective component to update at appropriate intervals all its corresponding subjects, and this information may be customised on a per-entity basis [Pet99, Mar02].

Figures 2 to 5 and Figure 7 illustrate a number of example applications, implemented during large European funded projects, using the Deva system. The Placeworld application (Figures 2 and 3) was a world containing worlds, each individual world being

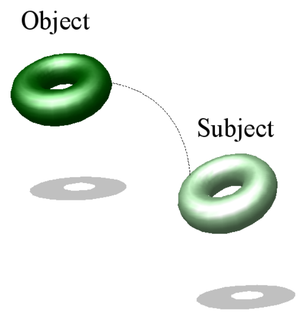


Figure 1: *The Deva object/subject model*

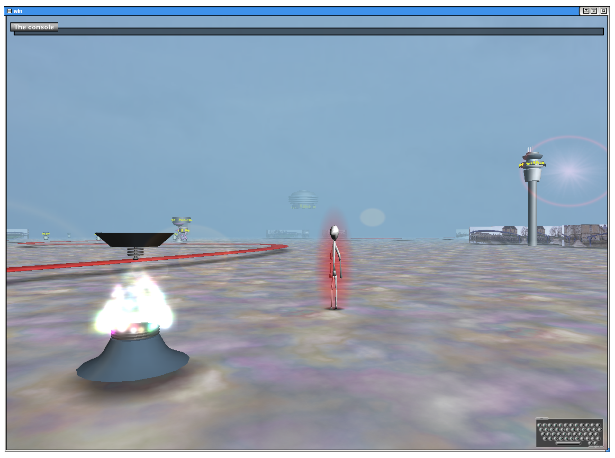


Figure 2: *Placeworld [PM01]*

an exhibit. It was developed for an art installation as part of a study into inhabited information spaces or electronic landscapes. Participants at the art show were able to explore each of the different exhibits, some of which had very different physical rules governing the behaviour of entities within them [PM01, Sha98, CP01]. They were also encouraged to contribute to the evolution of the landscape through the creation of hyper-links to favourite exhibits, and the use of personalised avatars [Mar02].

The QPit application [PCMT01] was implemented to visualise complex structured graph data, and employed a spring-mass-damper model to automatically configure its structure. By variations in the spring stiffness according to the different relationships between nodes (masses) in the data, the dynamics of the force model was used to determine an equilibrium configuration for the structure. This configuration presented a visualisation of the semantic information within the graph data.

The Senet Game (shown in Figure 5) was an educational distributed environment designed to give children experience of a board game found in the ruins of the ancient Egyptian city of Kahun [EPMW99], and through this, enable a study of how to better design social learning environments. The application used bespoke avatars (representing an adult, young girl and boy) together with a radiosity rendition of an Egyptian room, complete with the game board, movable pieces, die, and kartouches on the wall containing the game's instructions [Mar02]. Players took turns to throw the die and move their pieces according to the games rules; the aim being to be the first to remove all their pieces from the board.

The game itself was played by pairs of children, supervised in the environment by a teacher. A chat mechanism was used, which maintained a log (for later analysis) of all conversations that had taken place during a particular session of the game. Participants communicated by typing into an entry box that was part of the user interface. The game enabled participants to both co-operate to learn the rules, and compete to win [Mar02].

While a range of virtual reality applications do contain simulation and rich behaviour to varying degrees, these are commonly implemented as a part of bespoke applications. The Deva framework has demonstrated that with an appropriate infrastructure for complex behaviour supported as core functionality, it is possible to rapidly develop and customise the behaviour for a range

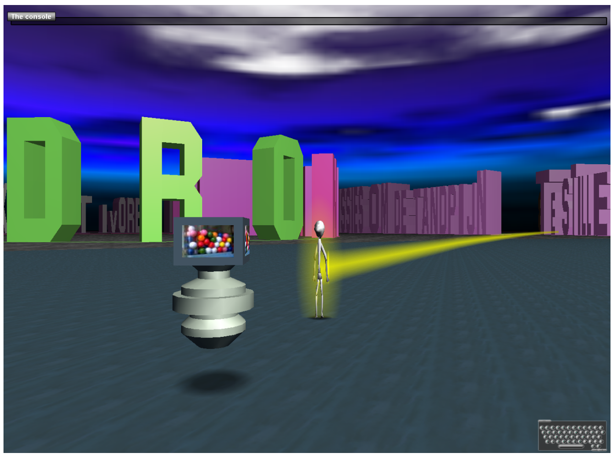
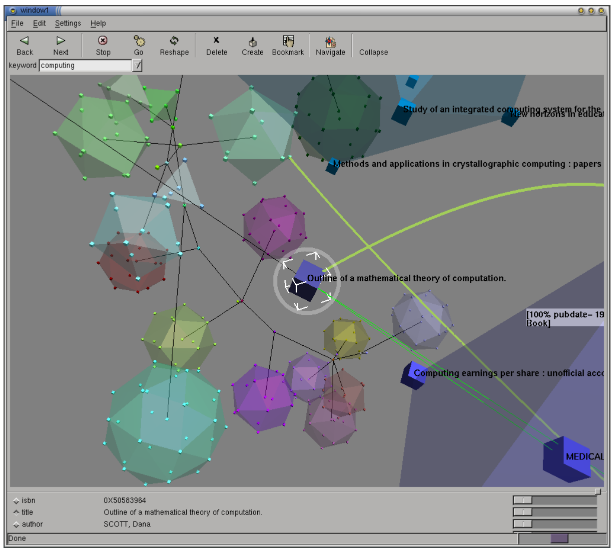


Figure 3: *Placeworld legible city [Sha98, CP01]*


window1
File Edit Settings Help
Back Next Stop Go Reshape Delete Create Bookmark Navigate Collapse
Study of an integrated computing system for the
New horizons in educat
Methods and applications in crystallographic computing : papers
Outline of a mathematical theory of computation.
[100% pubdate= 19
Book]
Computing earnings per share : unofficial acc
MEDICAL
Done

keyword computing

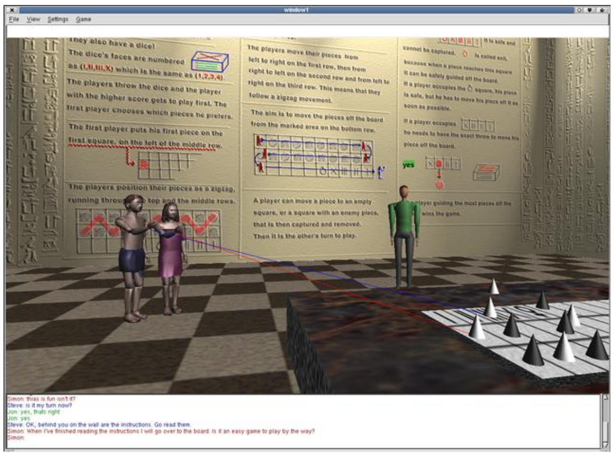
isbn 0X50583964
title Outline of a mathematical theory of computation.
author SCOTT, Dana

Figure 4: *QPit [PCMT01]*

of diverse applications. Physically based simulation techniques provide the most reliable method for computing realistic behaviour but often at a computational cost [Gle00]. Consequently these types of simulations are most often found in animation and computer aided design (CAD) applications, which impose less stringent real-time demands. Methods to manage the complexity of underlying simulations, while maintaining a consistent perception of the simulation offer the potential for wider scale adoption of simulation in real-time applications [GM98, GM99, GM00, GHP01].

3.1. Advantages

Rich behaviour in virtual environments can lead to highly engaging applications ranging from systems to prototype and maintain complex CAD designs, training, and entertainment applications. A particular example of a compelling and engaging application is VRSim's welding simulator (shown in Figure 6). Welding is an especially dangerous occupation and many trainees suffer serious burns during the early parts of their vocational training. VRSim's application aims to mitigate the level of injury through an immersive training setup prior to actual *hands-on* experience [VRS05]. The application combines tools very similar to real welding equipment. Users wear a custom head-mounted display which is fitted into a welders mask, graphical, auditory and



Simon: thias is fun isn't it?
Steve: is it my turn now?
Jon: yes, thats right
Jon: yes
Steve: OK, behind you on the wall are the instructions. Go read them.
Simon: When I've finished reading the instructions I will go over to the board. Is it an easy game to play by the way?
Simon:

Figure 5: *Senet game [EPMW99]*

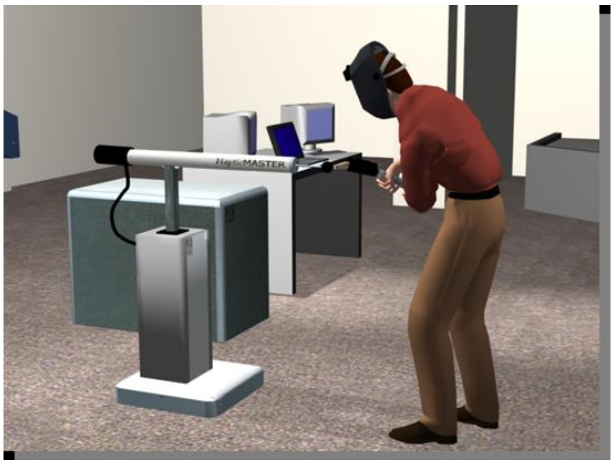


Figure 6: *Virtual welding simulator – Image courtesy of FCS Control Systems, The Netherlands*

haptic feedback (using an FCS HapticMASTER) is supplied to closely approximate multi-sensory informational cues in the real task. Good real-world correspondence between the training configuration, and the actual task may account for the good level of skills transfer reported by VRSim [Wal05]

3.2. Requirements

In order to maximise real world correspondence, and in doing so build compelling virtual reality applications, we argue that it is necessary to have the following:

- Good quality graphics
- Complex interaction preferably engaging multiple sensory modalities
- Realistic simulation

The level to which each of these factors are employed depends largely on the application. Good quality graphics may suffice for CAD applications, while high fidelity graphics may be required for interactive visualisations of architectural designs. Graphical and audio feedback may be sufficient for walk-throughs or simple visualisations, while haptic feedback may be required to assist in assembly tasks. Simple flocking behaviour [Rey87] may suffice for describing the behaviour of birds in synthetic landscapes, while more complex finite element models may be required to determine the deformation characteristics of materials in crash simulations [Win81, KSE*97, BFL*01].

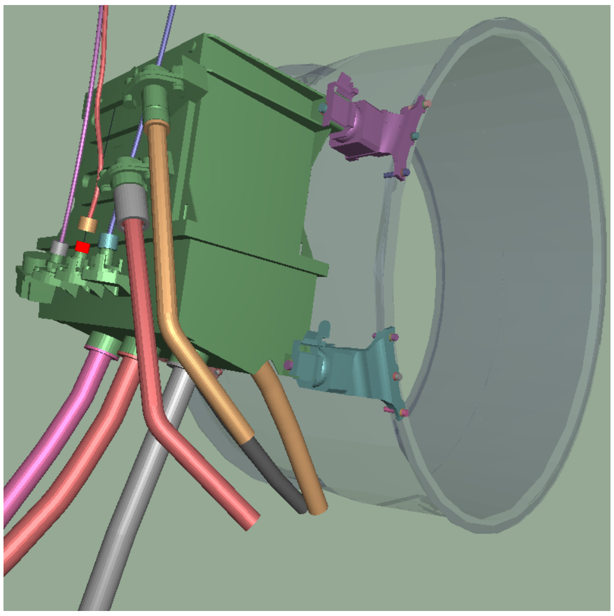


Figure 7: *Collaborative CAD prototyping application [MGP*04]*

With each of the above requirements for good real world correspondence imposing their own specific performance demands on applications, a particular challenge is to maximise the level of engagement within real-time performance constraints. Some of the techniques discussed in this tutorial may provide a mechanism for achieving this goal.

3.2.1. High Fidelity Rendering

High fidelity rendering is a term used to describe highly realistic graphical rendering. An important factor contributing to the realism of a computer generated image is the lighting. Illumination in the real-world is a complex process involving both natural and man-made light sources. We see shadows, reflections, optical effects caused by refraction, perspective effects, and we perceive depth through subtle visual cues. The weather affects visibility, clouds exhibit complex self shadowing and illumination characteristics. A huge range of techniques exist to simulate and compute realistic lighting effects [Gla89, Hec92, CWH93, SP94, Jen01, WKB*02, CDR02]. However, realistic illumination covering the full range of complex processes in the real world is still a significant challenge. Even more challenging is to achieve realistic illumination in real-time.

Few metrics for rendering fidelity exist and much of the current work has been carried out in immersive contexts [MTHC03, MR04]. Measures of rendering fidelity in these studies are normally arrived at by an analysis of the subjective preferences of participants. However, interestingly, these studies indicate that improved rendering fidelity could improve spatial task performance through better depth perception.

3.2.2. Multiple Interaction Modalities

The most common modality used to provide feedback in virtual environments is the visual one, colour changes are often used to indicate collisions or semantic information relating to tasks. Different graphical rendering styles can also be employed to convey information about entities in virtual environments. For example in a distributed CAD prototyping (Divipro) application, implemented using Deva, [MGP*04] and shown in Figure 7 a red highlight is used to visually indicate that an assembly constraint has been found. Part of the assembly is semi-transparent indicating it is semantically less important to the assembly task and merely provides contextual information.

Audio cues were used in this application to provide feedback relating to collisions, geometric constraints, task and collaboration state. A range of different tunes were played to indicate if a particular constraint had been matched, was active, was met or broken. Subtle changes in background colour were used to inform participants of transfer of control between them, in order to mediate collaboration. Informal observations of assembly sequence completion times showed that these cues improved the user's ability to perform assemblies. Since the application allowed multiple participants to collaborate on a given assembly, audio feedback was also provided to give information relating to allowable interactions, for example when it was acceptable for other users to partake in a sequence. Clearly audio cues could be incorporated into many virtual environment applications, to convey task related semantic information, but it is not the only alternative modality available.

The addition of haptic feedback offers a more engaging and tightly coupled interaction mechanism, in the sense that the same device is used both for input and display with each reinforcing the other. Due to limitations in both in the technology and software, most haptic devices currently available are unable to convey a real sense of touch as it is experienced in the real world. However, a sense of contact with solid or viscous materials (which resist to varying degrees) penetration can be conveyed. In the Divipro application, haptic feedback was used to provide force responses relating to collisions and locking to constraints. Again, informally, it was found to be a compelling addition to support performance of assembly tasks. Similarly, in the welding application, the use of haptic feedback conveys information about the position and material properties of the weld.

A final modality often exploited in re-creations of the past for museum exhibits (for example the Jorvik Viking Museum [Jor05]) is the sense of smell (or olfactory feedback). However, currently it is not practical to use this in virtual environments. For certain kinds of training tasks such as the welding simulator, it could be a useful modality to exploit as it would again convey additional information relating to the state of the material being used in the weld.

3.2.3. Multiple Participants

In the real world our senses are not only stimulated by interaction with entities in the environment, but also through interaction with other people. Multiple participants in virtual environments contribute to the realism and level of engagement an application can offer. Environments populated by synthetic characters soon become boring and predictable, as users quickly recognise any preprogrammed sequences. Real participants customise interactions in virtual environments through their own personal choices, and while these may be predictable (if you know the person) they are not always the same. Evidence for the suggestion that multiple participants improves the level of engagement in virtual environments can be found in the recent popularity of online multi-player gaming. In particular studies of communication between participants have shown that intuitive mechanisms improve the social experience of players and the level of engagement [HRF03]

In addition to communicating and interacting with each other, multiple participants can also choose to collaborate to achieve a shared objective. This phenomenon is also seen in multi-player online games, where friends collaborate to ‘take out’ participants they do not consider to be in their social group [Man01b]. In a number of training and assembly or maintenance tasks, collaborative interaction may also be essential to completing the task. However in a shared behaviour-rich environment it is essential that each participant sees a plausible representation of the environment, and that the consistency of this is always maintained [Pet99, Mar02]. As we will explain in this tutorial, this can be non-trivial.

3.2.4. Complex Simulation

Complex simulations that application developers may typically wish to incorporate into virtual environments can be classified as follows:

- Emergent behaviour
- Material properties
- Natural phenomena

Emergent behaviour involves a number of mainly independent entities each conforming to some simple particle physics, scripted responses, or ad-hoc goals. Combined, the behaviour of these entities leads to a complex overall response. This type of simulation is used to support flocking behaviour [Rey87, BCN97], and crowd simulations in multi-user [BGL97, MBCT98] and evacuation scenarios [UT01, UT02].

More complex numerical simulation techniques are used to compute material properties and motion of particles, rigid-body, articulated-rigid-body, and deformable systems. Often these employ numerical solvers to solve equations of motion and deformation characteristics of finite elements, or surface fitting methods to compute iso-surfaces from force field functions (implicit surfaces [BW97]). These techniques are commonly used in animations for example, to simulate soft substances [DG95, CGD97], deformable objects [TF88, TF98], cloth draping behaviour [BHW94, BW98, BWK03], and in applications such as crash [Win81, KSE*97, EMTTT98, BFL*01] and surgical simulations [BN98, CDA99, DCA99, BHS01, WDGT01]. However, the performance demands of many of these methods have meant that few have found use in many real-time virtual reality applications.

Simulating natural phenomena such as wind, smoke, fire [SF93, Sta00, FSJ01], clouds [Bli82, MYDN01], aurora [BRS*03] and water [FR86, KWF*01, EMF02] is an active research area and a large variety of traditional procedural methods [MPPW94], fluid dynamic simulations, and modern implementations employing the graphical processor unit (GPU) exist [HCSL02, HL01, HBSL03, Har04, LLW04]. Many of these methods achieve visually impressive results in real-time, with research into faster and more realistic simulations ongoing. Illumination techniques have also been developed to incorporate optical effects such as rainbows, cloud-bows, sun-dogs [Mus89], atmospheric mirages [KH97] and many other subtle lighting effects commonly seen in the real world.

However combining simulation, rendering, realistic illumination, complex interaction, and multiple participants in real-time environments is a significant computational challenge. Techniques to selectively compute and render perceptually important parts of the environment, by exploiting the effects of high cognitive load during task performance, is essential to achieving this in real-time. In the following sections, we will elaborate on this.

4. Perceptually Based Graphical Rendering

The computer graphics industry, and in particular those involved with films, games, simulation and virtual reality, continue to demand more realistic computer generated images, that is synthesised images that more accurately match the real scene they are intended to represent. Despite the ready availability of modern high performance graphics cards, the complexity of the scenes being modelled and the high fidelity required of the images means that rendering such images is still simply not possible in a reasonable, let alone real-time on a single computer. Two approaches may be considered in order to achieve such realism in real-time: Parallel Processing and Visual Perception guided methods. Parallel Processing has a number of computers working together to render a single image, which appears to offer almost unlimited performance, however, enabling many processors to work efficiently together is a significant challenge [CDR02, PMS*99, WSBW01]. Visual Perception, on the other hand, takes into account that it is the human who will ultimately be looking at the resultant images, and while the human eye is good, it is not perfect. As we will see in this section, exploiting knowledge of the human visual system can indeed save significant rendering time by simply not computing detail in those parts of a scene that the human will fail to notice.

4.1. High Fidelity Graphics

Realness - the state of being actual or real. Obviously this definition refers to the “real” world and our perception of it, however frequently in the doctrine of computer science the terms “realistic”, “realism” and “real” are discussed. Obviously anything represented on a computer is not real but just an approximation, so what do these expressions refer to? There are many uses for computers in the world we live in ranging from high performance games to high accuracy mathematical calculations. Both of these examples and countless more have one thing in common the need to have some level of realism. Within the games industry it is important for there be some link with reality (or at least some conceivable fantasy of reality) to involve the player in the game. However the level of realism needed in a computer game is related to the genre and objective of the game. At the other end of the spectrum there exist applications that directly apply to the real world; one example might be a software package that is employed to perform aerodynamics calculations during the design process of a new fighter aircraft. In this circumstance an extremely high fidelity simulation of reality is required to ensure that the plane will fly. However within the context of a computer game it is more important that the plane looks realistic and behaves as expected, while in the design application the appearance of the plane is less critical (and may not even be presented) but realistic behaviour is crucial to the application.

4.2. Image Quality Metrics

Reliable image quality assessments are necessary for the evaluation of realistic image synthesis algorithms. Typically the quality of the image synthesis method is evaluated using image-to-image comparisons. Often comparisons are made with a photograph of the scene that the image depicts. Several image fidelity metrics have been developed whose goals are to predict the amount of differences that would be visible to a human observer. It is well established that simple approaches like mean squared error do not provide meaningful measures of image fidelity, thus more sophisticated measures which incorporate a representation of the human visual system are needed. It is generally recognised that more meaningful measures of image quality are obtained using techniques based on visual (and therefore subjective) assessment of images, after all most final uses of computer generated images will be viewed by human observers.

4.2.1. Perceptually Based Image Quality Metrics

A number of experimental studies have demonstrated many features of how the human visual system works. However, problems arise when trying to generalise these results for use in computer graphics. This is because, often, experiments are conducted under limited laboratory conditions and are typically designed to explore a single dimension of the human visual system. Instead of reusing information from these previous psychophysical experiments, new experiments are needed which examine the human visual system as a whole rather than trying to probe individual components. Using validated image models that predict image fidelity, programmers can work toward achieving greater efficiencies in the knowledge that resulting images will still be faithful visual representations. Also in situations where time or resources are limited and fidelity must be traded off against performance, perceptually based error metrics could be used to provide insights into where corners could be cut with least visual impact. Using a simple five sided cube as their test environment Meyer et al. [MRC*86] presented an approach to image synthesis comprising separate physical and perceptual modules. They chose diffusely reflecting materials to build a physical test model. Each module was verified using experimental techniques. The test environment was placed in a small dark room. Radiometric values predicted using a radiosity lighting simulation were compared to physical measurements of the radiant flux density in the real scene. Results showed that irradiation was greatest near the centre of the open side of the cube. This area provided the best view of the light source and other walls. In summary, there was a good agreement between the radiometric measurements and the predictions of the lighting model.

Rushmeier et al. [RLP*95] explored using perceptually based metrics, based on image appearance, to compare image quality to a captured image of the scene being represented. The goal of this work was to obtain results by comparing two images using models that give a large error when differences exist between images. The following models attempt to model effects present in the human visual system. Each uses a different Contrast Sensitivity Function (CSF) to model the sensitivity to spatial frequencies.

Model 1 After Mannos and Sakrison: First, all the luminance values are normalised by the mean luminance. The non linearity in perception is accounted for by taking the cubed root of each normalised luminance. A Fast Fourier Transform (FFT) is computed of the resulting values, and the magnitudes of the resulting values are filtered with a CSF to an array of values. Finally the distance between the two images is computed by finding the Mean Square Error (MSE) of the values for each of the two images. This technique therefore measures similarity in Fourier amplitude between images.

Model 2 After Gervais et al: This model includes the effect of phase as well as magnitude in the frequency space representation of the image. Once again the luminances are normalised by dividing by the mean luminance. An FFT is computed producing an array of phases and magnitudes. These magnitudes are then filtered with an anisotropic CSF filter function constructed by fitting splines to psychophysical data.

Model 3 After Daly: In this model the effects of adaptation and non-linearity are combined in one transformation, which acts on each pixel individually. In the first two models each pixel has significant global effect in the normalisation by contributing to the image mean. Each luminance is transformed by an amplitude nonlinearity value. An FFT is applied to each transformed luminance and then they are filtered by a CSF (computed for a level of 50 cd/m2). The distance between the two images is then computed using MSE as in model 1.

The Visible Difference Predictor (VDP) is a perceptually based image quality metric proposed by Daly [Dal93]. Myskowski realised this metric had many potential applications in realistic image synthesis [Mys98]. He completed a comprehensive validation and calibration of VDP response via human psychophysical experiments. The VDP was tested to determine how close predictions come to subjective reports of visible differences between images by designing two human psychophysical experiments. Results from these experiments showed a good correspondence for shadow and lighting pattern masking and in comparison of the perceived quality of images generated as subsequent stages of indirect lighting solutions.

4.3. Low-Level Perception-Based Error Metrics

Perceptual error metrics have also been used in several other areas. Gibson and Hubbold [GH97b] proposed a perception-driven hierarchical algorithm for radiosity used to decide when to stop hierarchy refinement. Links between patches are not re-fined anymore once the difference between successive levels of elements becomes unlikely to be detected perceptually. Gibson and Hubbold also applied a similar error metric to measure the perceptual impact of the energy transfer between two interacting patches, and to decide upon the number of shadow feelers that should be used in visibility test for these patches. Perceptually-informed error metrics have also been successfully introduced to control the adaptive mesh subdivision and mesh simplification. Specific implementations have been performed and analysed by Myszkowski [Mys98], Gibson et al. [GCHH03] 28, and Volevich et al. [VMKK00].

4.3.1. Advanced Perception-Based Error Metrics

The scenario of embedding advanced human visual system models into global illumination and rendering algorithms is very attractive, because computation can be perception-driven specifically for a given scene. Bolin and Meyer [BM98] developed an efficient approximation of the Sarnoff Visual Discrimination Model (VDM), which made it possible to use this model to guide samples in a rendered image. Because samples were only taken in areas where there were visible artifacts, some savings in rendering time compared to the traditional uniform or adaptive sampling were reported. Myszkowski [Mys98] has shown some applications of the VDP to drive adaptive mesh subdivision taking into account visual masking of the mesh-reconstructed lighting function by textures. Ramasubramanian et al. [RPG99] have developed their own image quality metric which they applied to predict the sensitivity of the human observer to noise in the indirect lighting component. This made possible more efficient distribution of indirect lighting samples by reducing their number for pixels with higher spatial masking (in areas of images with high frequency texture patterns, geometric details, and direct lighting variations). All computations were performed within the framework of the costly path tracing algorithm, and a significant speedup of computations was reported compared to the sample distribution based on purely stochastic error measures. A practical problem arises that the computational costs incurred by the human visual system models introduce an overhead to the actual lighting computation, which may become the more significant the more rapid is the lighting computation. This means that the potential gains of such perception-driven computation can be easily cancelled by this overhead depending on many factors such as the scene complexity, performance of a given lighting simulation algorithm for a given type of scene, image resolution and so on. The human visual system models can be simplified to reduce the overhead, e.g., Ramasubramanian et al. [RPG99] ignore spatial orientation channels in their visual masking model, but then underestimation of visible image artifacts becomes more likely. To prevent such problems and to compensate for ignored perceptual mechanisms, more conservative (sensitive) settings of the human visual system models should be applied, which may also reduce gains in lighting computation driven by such models.

4.3.2. Visible Differences Predictor

Although, substantial progress in physiology and psychophysics studies has been achieved in recent years, the human visual system as the whole, and in particular, the higher order cognitive mechanisms, are not fully understood. Only the early stages of the visual pathway beginning with the retina and ending with the visual cortex are considered as mostly explored. It is believed that the internal representation of an image by cells in the visual cortex is based on spatial frequency and orientation channels. The channel model provides a good explanation of visual characteristics such as:

- The overall behavioural Contrast Sensitivity Function (CSF) - visual system sensitivity is a function of the spatial frequency and orientation content of the stimulus pattern.
- Spatial masking - detect ability of a particular pattern is reduced by the presence of a second pattern of similar frequency content.
- Sub-threshold summation - adding two patterns of sub-threshold contrast together can improve detect ability within a common channel.
- Contrast adaptation - sensitivity to selected spatial frequencies is temporarily lost after observing high contrast patterns of the same frequencies.
- The spatial frequencies after effects - as result of the eye adaptation to a certain grating pattern, other nearby spatial frequencies appear to be shifted.

Because of these favourable characteristics, the channel model provides the core of the most recent human visual system models that attempt to describe spatial vision. The VDP is considered one of the leading computational models to predicting the differences between images that can be perceived by the human observer. The VDP receives as input a pair of images, and as output it generates a map of probability values, which characterise perceptibility of the differences. The input target and mask images undergo an identical initial processing, as shown in Figure 8. At first, the original pixel intensities are compressed by the amplitude non-linearity based on the local luminance adaptation, simulating Weber's law-like behaviour. Then the resulting image is converted into the frequency domain and processing of CSF is performed.

The resulting data is decomposed into the spatial frequency and orientation channels using the Cortex Transform, which is a pyramid-style, invertible, and computationally efficient image representation. Then the individual channels are transformed back to the spatial domain, in which visual masking is processed. For every channel and for every pixel, the elevation of detection threshold is calculated based on the mask contrast for that channel and that pixel. The resulting threshold elevation maps can be computed for the mask image, or mutual masking can be considered by taking the minimal threshold elevation value for the corresponding channels and pixels of the two input images. These threshold elevation maps are then used to normalise the contrast differences between target and mask images. The normalised differences are input to the psychometric function which estimates probability of detecting the differences for a given channel. This estimated probability value is summed across all channels for every pixel. Finally, the probability values are used to visualise visible differences between the target and mask images. It is assumed that the difference can be perceived for a given pixel when the probability value is greater than 0.75, which is standard threshold value for discrimination tasks. When a single numeric value is needed to characterise the differences between images, the percentage of pixels with probability greater than this threshold value is reported. The main advantage of the VDP is a prediction of local differences between images (on the pixel level). The original Daly model also has some disadvantages, for example, it does not process chromatic channels in input images. However, in global illumination applications many important effects such as the solution convergence or the quality of shadow reconstruction can be relatively

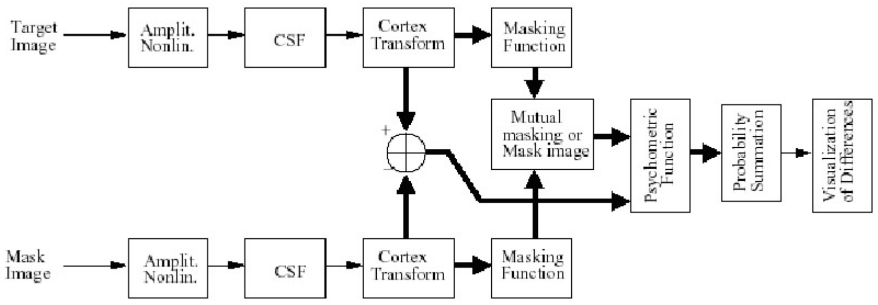


Figure 8: *Block diagram of the Visible Differences Predictor (heavy arrows indicate parallel processing of the spatial frequency and orientation channels)*

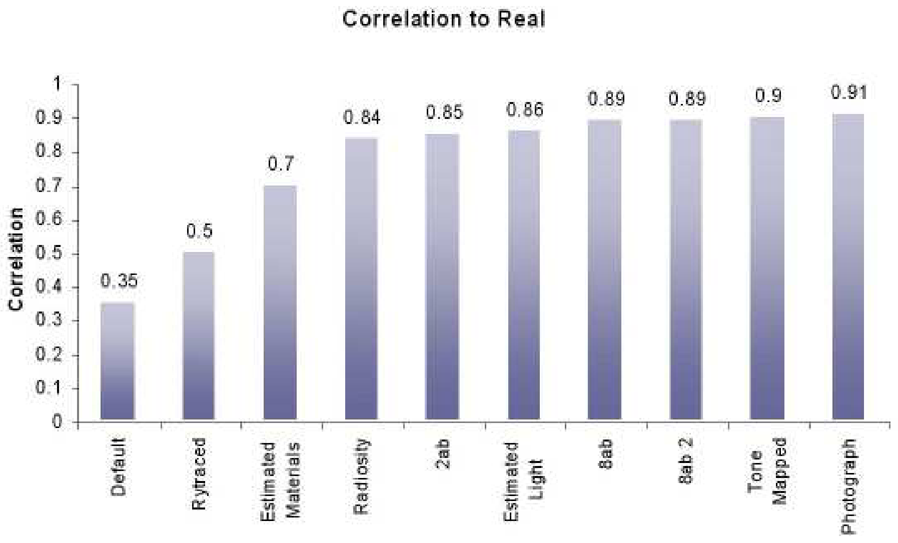


Figure 9: *Results obtained by McNamara et al. in lightness matching task experiments*

well captured by the achromatic mechanism, which is far more sensitive than its chromatic counterparts. The VDP seems to be one of the best existing choices for the prediction of image quality for various settings of global illumination solutions.

4.4. Comparing Real and Synthetic Images

A number of experiments have been conducted at the University of Bristol where comparisons have been made between real and synthetic images. These comparisons although comparing real and synthetic images have been task specific and have employed only simple controlled environments. McNamara [MCTG00], performed a series of experiments where subjects were asked to match lightness patches within the real world to those on a VDU. They discovered that a photograph of the real scene gave the highest perceptual match, with a high quality tone mapped rendered version coming a close second. A graph of their findings is shown in Figure 9. In all cases (apart from the ray tracing and radiosity results) Radiance was used to render the images.

4.5. Selective Rendering

The perception of a virtual environment depends on the user and where he/she is currently looking in that environment. Visual attention is the process by which we humans select a portion of the available visual information for localisation, identification and understanding of objects in an environment. It allows our visual system to process visual input preferentially by shifting attention about an image, giving more attention to salient locations and less attention to unimportant regions [Dal93, IKN98, YPG01]. When attention is not focused onto items in a scene they can literally go unnoticed [CCL02, Yar67]. So whilst visual attention may not be appropriate for, for example, a completed computer generated film which will be watched by many viewers simultaneously, it can certainly assist in the production of the film when the developers are focusing on particular aspects of a scene, for example how the movement of a character affects the lighting in a particular region.

The key to achieving realism in real-time in virtual environments on current technology is knowing where the user will be looking in the image and rendering these areas at a very high quality, while the remainder of the scene, not attended to by the user, can be rendered to a much lower quality without the user being aware of the quality difference [YPG01, CCL02, CKM03]. For a surprisingly large number of applications, high level task maps and low level saliency maps can indeed indicate where the user will be looking with complete accuracy. We thus define a perceptually realistic scene as one in which the viewer perceives to be of a very high quality, where in fact significant parts of the image may be rendered at a much lower quality.

4.5.1. Visual Perception

Visual attention is a coordinated action involving conscious and unconscious processes in the brain, which allow us to find and focus on relevant information quickly and efficiently. If detailed information is needed from many different areas of the visual environment, the eye does not scan the scene in a raster-like fashion, but jumps so that the relevant objects fall sequentially on the fovea. These jumps are called saccades [Yar67].

There are two general visual attention processes, labelled bottom-up and top-down, which determine where humans locate their visual attention [Jam90]. The bottom-up process is purely stimulus driven, for example, a fire in the dark, a sudden movement, a red apple in a green tree, or the lips and eyes of another person - the most mobile and expressive elements of a face. In all these cases, the visual stimulus captures attention automatically without volitional control. This is evolutionary; the movement may be danger lurking behind a bush, or we may need to find ripe fruit for our meal. In contrast, the top-down process is under voluntary control, and focuses attention on one or more objects that are relevant to the observer's goal when studying a scene. Such goals might include looking for a lost child, searching for an exit, or remembering the position of the people and objects in a room.

General knowledge of the human visual system has been widely used to improve the quality of rendered images [FPSG96, GTS*97, MCTG00, MTAS01, PFG98, RPG99]. Other approaches have focused on how complex model detail can be reduced without any reduction in the viewer's perception of the environment [LH01, LRW*01, MS95, Red97, BFM01].

The application of visual attention models in computer graphics has so far mainly exploited only peripheral vision and the bottom-up visual attention process. Recent approaches however have combined both the top-down and bottom-up processes [SDL*05].

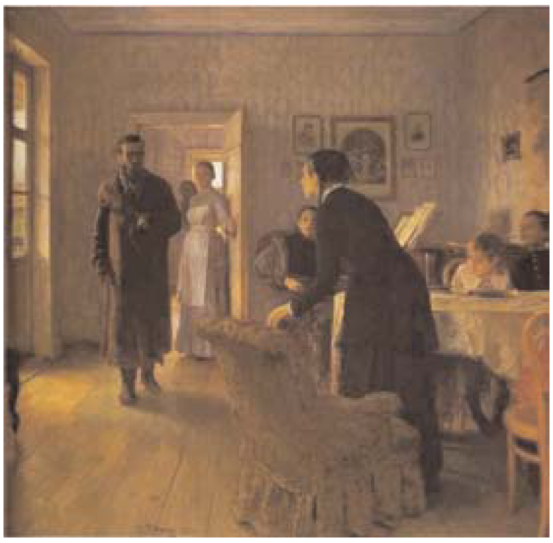
4.5.2. Peripheral Vision

Due to the fact that the human eye only processes detailed information from a relatively small part of the visual field, it is possible to reduce detail in the periphery without upsetting visual processing. In numerous studies, McConkie and Loschky [ML97, LM99, LMYM01] used an eye-linked, multiple resolution display that produces high visual resolution only in the region to which the eyes are directed. They were able to show that photographic images filtered with a window radius of 4.1ř produced results statistically indistinguishable from that of a full, high-resolution display. The display they propose does, however, encounter the problem of updating the multi-resolution image after an eye movement without disturbing the visual processing. Their work has shown that the image needs to be updated after an eye saccade within 5 milliseconds of a fixation, otherwise the observer will detect the change in resolution. These high update rates were only achievable using an extremely high temporal resolution eye tracker, and pre-storing all possible multi-resolution images that were to be used.

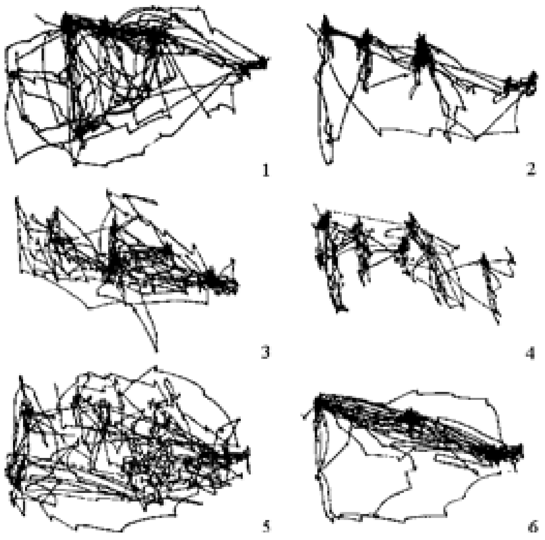
In another experiment, Watson et al. [1997] evaluated the effectiveness of high detail insets in head-mounted displays. The high detail inset they used was rectangular and was always presented at the finest level of resolution. Three inset conditions were investigated: a large inset - half the complete display's height and width, a small inset size - 30% of the complete display's height and width, and no inset at all. The level of peripheral resolution was varied at: fine resolution 320×240, medium resolution 192×144 and coarse resolution 64×48. Their results showed that although observers found their search targets faster and more accurately in a full high resolution environment, this condition was not significantly better than the high-resolution inset displays with either medium or low peripheral resolutions.

4.5.3. Inattentional Blindness

In 1967, the Russian psychologist Yarbus recorded the fixations and saccades observers made while viewing natural objects and scenes. Observers were asked to answer different questions concerning the scene in Repin's depiction of 'An Unexpected



(a) Repin's picture



(b) Eye movements

Figure 10: *The effects of a task on eye movements. Repin's picture was examined by subjects given one of six different instructions; 1. Free viewing, 2. Judge their ages, 3. Guess what they had been doing before the unexpected visitor's arrival, 4. Remember the clothes worn by the people, 5. Remember the position of people and objects in the room & 6. Estimate how long the unexpected visitor has been away from the family [Yar67]*

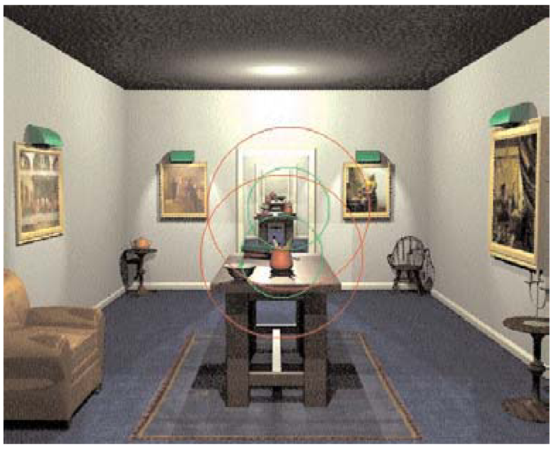
Visitor' [Yar67]. This resulted in substantially different saccade patterns, each one being easily construable as a sampling of those picture objects that were most informative for the answering of the questions, as shown in Figure 10.

The failure of the human to see unattended items in a scene, is known as inattentional blindness [MR98,SC99]. The concept of task maps, which are two dimensional maps highlighting the task at hand, has recently been introduced to exploit this top-down approach [CCW03].

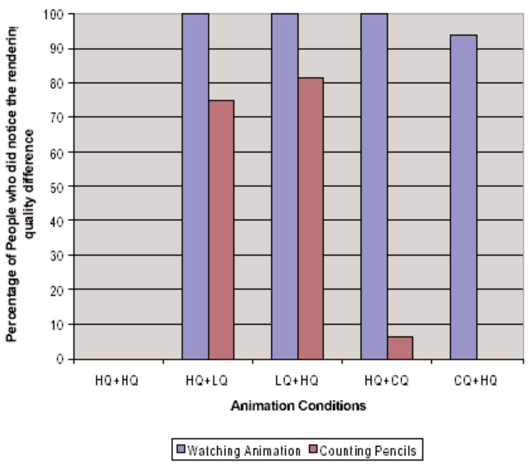
Previously, Cater et al. [CCL02] showed that the conspicuous objects in a scene that would normally attract the viewer's attention can be deliberately ignored if they are irrelevant to the task at hand. The effectiveness of inattentional blindness in reducing overall computational complexity was investigated by asking a group of users to perform a specific task: to watch two animations and in each of the animations, count the number of pencils that appeared in a mug on a table in a room as he/she moved on a fixed path through four such rooms. In order to count the pencils, the users needed to perform a smooth pursuit eye movement tracking the mug in one room until they have successfully counted the number of pencils in that mug and then perform an eye saccade to the mug in the next room. The task was further complicated and thus retained the viewer's attention, by each mug also containing a number of spurious paintbrushes. The study involved three rendered animations of an identical fly through of four rooms. The only difference being the quality to which the individual animations had been rendered. The three qualities of animation were:

High Quality(HQ): Entire animation rendered at the highest quality.
Low Quality(LQ): Entire animation rendered at a low quality with no anti-aliasing.
Circle Quality(CQ): Low Quality Picture with high quality rendering in the visual angle of the fovea (2 degrees) centred around the pencils, shown by the inner green circle in Figure 11(a). The high quality is blended to the low quality at 4.1 degrees visual angle (the outer red circle in figure 11(a)).

Each frame for the high quality animation took on average 18 minutes 53 seconds to render on a Intel Pentium 4 1GHz Processor, while the frames for the low quality animation were each rendered on average in only 3 minute 21 seconds. A total of 160 subjects were studied which each subject seeing two animations of 30 seconds each displayed at 15 frames per second. Fifty percent of the subjects were asked to count the pencils in the mug while the remaining 50% were simply asked to watch



(a) Visual angle covered by the fovea for mugs in the first two rooms at 2 degrees (green circles) and 4.1 degrees (red circles)



(b) Experimental results for the two tasks: Counting the pencils and simply watching the animations

Figure 11: *Results of an experiment to assess the range of the region of interest for high quality rendering while performing a task and observing and animation*

the animations. To minimise experimental bias the choice of condition to be run was randomised and for each, 8 were run in the morning and 8 in the afternoon. Subjects had a variety of experience with computer graphics and all exhibited at least average corrected vision in testing. A count down was shown to prepare the viewers that the animation was about to start followed immediately by a black image with a white mug giving the location of the first mug. This ensured that the viewers focused their attention immediately on the first mug and thus did not have to look around the scene to locate it. On completion of the experiment, each participant was asked to fill in a detailed questionnaire. This questionnaire asked for some personal details, including age, occupation, sex and level of computer graphics knowledge. The participants were then asked detailed questions about the objects in the rooms, their colour, location and quality of rendering. These objects were selected so that questions were asked about objects both near the foveal visual angle (located about the mug with pencils) and in the periphery. They were specifically asked not to guess, but rather state ‘don’t remember’ when they had failed to notice some details.

Figure 3(b) shows the overall results of the experiment. Obviously the participants did not notice any difference in the rendering quality between the two HQ animations (they were the same). Of interest is the fact that, in the CQ + HQ experiment, 95% of the viewers performing the task consistently failed to notice any difference between the high quality rendered animation and the low quality animations where the area around the mug was rendered to a high quality. Surprisingly 25% of the viewers in the HQ+LQ condition and 18% in the LQ+HQ case were so engaged in the task that they completely failed to notice any difference in the quality between these very different qualities of animation. Furthermore, having performed the task of counting the pencils, the vast majority of participants were simply unable to recall the correct colour of the mug (90%) which was in the foveal angle and even less the correct colour of the carpet (95%) which was outside this angle. The ‘failure to notice’ was even higher for ‘less obvious’ objects, especially those outside the foveal angle. Overall the participants who simply watched the animations were able to recall far more detail of the scenes, although the generic nature of the task given to them precluded a number from recalling such details as the colour of specific objects, for example 47.5% could not recall the correct colour of the mug and 53.8% the correct colour of the carpet.

The results of this work demonstrated that when observers were performing the task within an animation, their visual attention was fixed exclusively on the area of the task and they consistently failed to notice the significant difference in rendering quality between the two animations. Of course, knowledge of the task being performed was fundamental in determining where a view was attended. For many applications, for example, film production, games and simulators, such knowledge exists.

4.5.4. Task and Saliency Maps

Low-level saliency models determine what visual features will involuntarily attract our attention in a scene. Visual psychology researchers such as Yarbus [Yar67], Itti and Koch [IK00] and Yantis [Yan96] showed that the visual system is highly sensitive to features such as edges, abrupt changes in colour, and sudden movements. This low-level visual processing has been exploited in computer graphics by Yee et al. [Yee00, YPG01] to accelerate animation renderings with global illumination, by applying a model of visual attention to identify conspicuous regions. Yee constructs a spatiotemporal error tolerance map, called the Aleph map, from spatiotemporal contrast sensitivity and a low-level saliency map, for each frame in an animation. The saliency map is obtained by combining the conspicuity maps of intensity, colour, orientation and motion. Subsequent work by Marmitt and Duchowski [MD02] showed, however, that care must be taken with bottom-up visual attention models as they do not always predict attention regions in a reliable manner.

4.5.5. Importance Maps

Sundstedt el al [SDL*05] have a developed a rendering framework that exploits visual attention processes, in order to selectively render high-fidelity animations in a reasonable time, while maintaining perceived quality.

The framework is composed of two major processes:

- selective guidance uses a combination of saliency and a measure of task relevance to direct the rendering computation.
- selective rendering corresponds to the traditional rendering computation.

However, computational resources are focused on parts of the image which are deemed more important by the selective guidance.

4.5.6. Selective Guidance System

Sundstedt et al.'s selective guidance system produces an importance map which is used to direct the rendering process. An importance map, *IM(wt,ws,op)*, is a two-dimensional map representing image space. The values within the map dictate where computational resources are best spent to obtain the highest perceptual result, whereby the highest values will result in preferential rendering. The importance map is a combination of a task map to model the effect of top-down visual attention and a saliency map to model bottom-up visual attention. The parameters in the importance map allow the user to specify a weighting that defines the relative importance of the task map and the saliency map. The parameter *wt* is a coefficient which is applied to the values in the task map. The other coefficient *ws* is applied to the values in the saliency map, and the two terms are combined through the operator *op*.

Selection of an appropriate operator controls the combination of the two maps in the selective renderer. The implementation currently uses addition to combine the information from both maps such that all weighted features are preserved.

4.5.7. Map Generation

The process begins with a rapid image estimate (in the order of ms) of the scene using a quick rasterisation pass in hardware [LDC05]. This estimate can be used in two ways. Firstly for building the task map by identifying user-selected task objects, and secondly, by using it as an input to a saliency generator. In the creation of the task map the program reads in the geometry information and a list of predefined task objects. It then produces a map with task objects in white and the other geometry in black. A foveal-angle gradient is then applied around task-related objects in the task map [SCCD04]. In the creation of the saliency map the image estimate serves to locate areas where an observer will be most likely to look.

Figure 12 shows the various maps for the reference image 14(a). Sub-figures 12(a) and 12(b), in Figure 12 show the task map with and without a foveal-angle gradient added. Sub-figure 12(c) demonstrates the saliency map, while 12(d) shows the combination of the task map and the saliency map with equal weighting.

4.5.8. Results

To test their importance maps, Sundstedt et al. rendered ten animations of a corridor scene [SDL*05] shown in Figure 14. They rendered a high quality (HQ), low quality (LQ), task map quality (TQ), saliency map quality (SQ) and a combined task map and saliency map quality (IQ) animation. Table 1 shows the conditions tested and some of the maximum and minimum rendering parameters values used for the renderings. The animations were all rendered at 900×900 resolution. Figure 13 shows the timing comparison between the reference high quality solution (HQ) and the different conditions generated using the importance map. Calculations were done on an Intel Xeon processor running at 2.4 GHz with 3 GB of memory under Linux. Rendering the entire

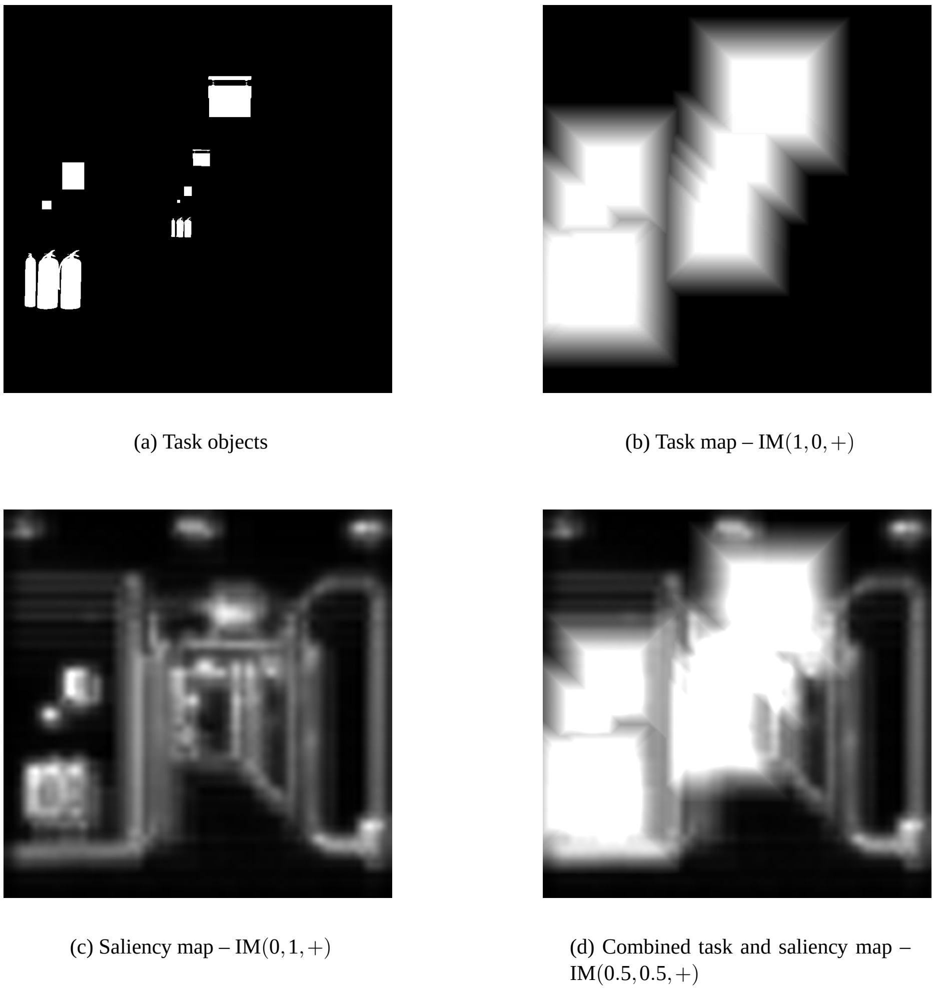

(a) Task objects
(b) Task map – IM(1,0,+)
(c) Saliency map – IM(0,1,+)
(d) Combined task and saliency map – IM(0.5,0.5,+)

Figure 12: *Importance map examples from the corridor scene*

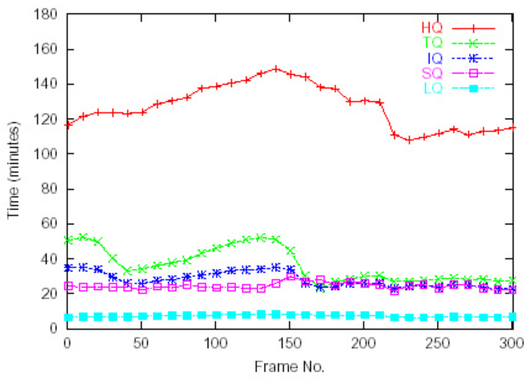


Figure 13: *Timing comparison for corridor A between the reference high quality solution (HQ) and the different conditions generated using the* importance map

Acronym	Description
HQ	High Quality: all pixels rendered using max settings (rp=16, st=0.01)
LQ	Low Quality: all pixels rendered using min settings (rp=1, st=1)
TQ	Task Quality:selectively rendered using only task map as input (IM(1,0,+)
SQ	Saliency Quality:selectively rendered using a saliency map as input (IM(0,1,+)
IQ	Task Quality:selectively rendered using saliency map and task map combined as input (IM(0.5,0.5,+)

Table 1: *Animation pairs shown in the experiment: (a) HQ/HQ, (b) HQ/LQ, (c) LQ/HQ, (d) HQ/TQ, (e) TQ/HQ, (f) HQ/SQ, (g) SQ/HQ, (h) HQ/IQ, and (i) IQ/HQ. (rp = rays per pixel, st = specular threshold.)*

frame to the same detail as the task objects in Sub-figure 12(a) takes 5.2 times longer than the optimised method using only the saliency map, 3.5 with the task map and 4.5 longer than when using the task map and saliency map combined.

Sundstedt et al. then went on to show, through a series of psychophysical experiments, that when performing a task in the environment (checking the fire safety equipment), the viewers were unable to notice a difference between the high quality rendered animation, and the selective quality animation, which was rendered at a fraction of the computational cost [SDL*05].

4.6. Perceptual Realism in Real-Time

As we have seen, visual perception, including saliency and inattentional blindness can in fact be exploited to significantly reduce the rendered quality of a large portion of a scene without having any affect on the viewer's perception of the scene. This knowledge will enable us to prioritise the order and the quality to which any pixel in our image should be rendered. This does, of course, depend on the scene and knowledge of the task being performed. For many applications, for example games and simulators, such knowledge exists offering the real potential of using visual perception approaches to achieve 'perceptually realistic' rendering in real-time.

Acknowledgements

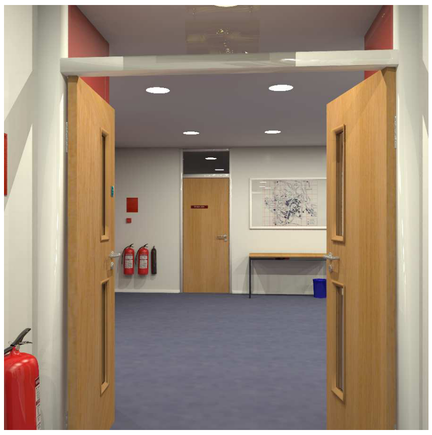
We would like to gratefully acknowledge the contributions of Alan's students in particular Veronica Sundstedt who has kindly provided much of the material on the selective guidance system and Importance Maps.

(a) Corridor A: The first frame

(b) Corridor A: The 156th frame

(c) Corridor A: The final frame

(d) Corridor B: The first frame


(e) Corridor B: The 156th frame

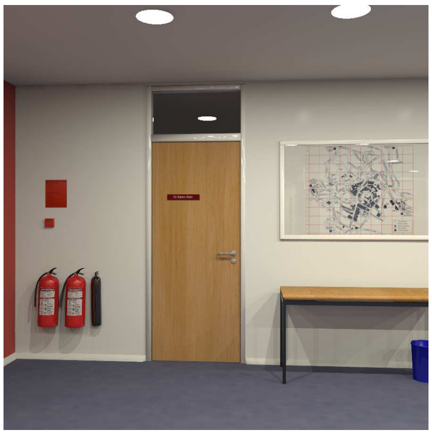

(f) Corridor B: The final frame

Figure 14: *Corridors A and B from a 312 frame animation*

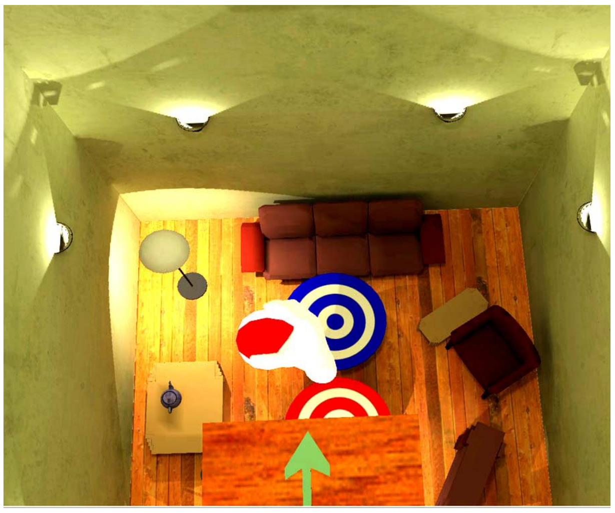


Figure 15: *The UNC virtual pit experiment [MRWJ03]*

5. Interaction in Distributed Environments

A particularly compelling example of a virtual environment, shown in Figure 15, is the University of North Carolina's 'Pit' model [MIWJ02, MRWJ03]. Although the environment is rendered with relatively low graphical fidelity (by modern standards), a number of multi-sensory cues contribute to its overall richness. These include a telephone that rings, a radio playing music, an open window with a curtain blowing in the breeze, a carefully positioned fan to give the impression of that breeze, and passive haptics consisting of styrofoam blocks placed where virtual entities such as the walls of the room are located. Combined with a set of stressful tasks to perform, participants report a compellingly engaging experience.

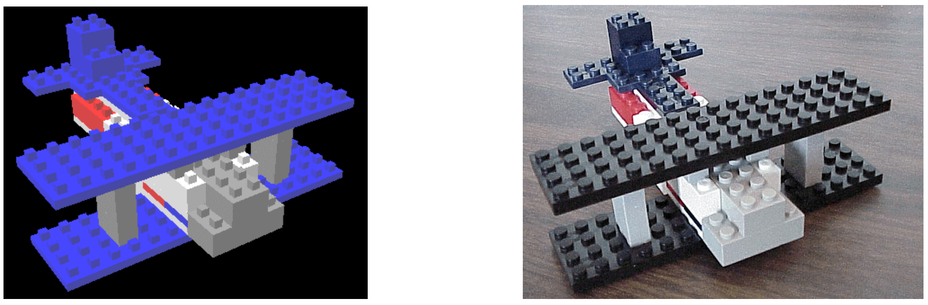
As the field of virtual reality matures, our expectations of the tasks that can be carried out, and the behavioural fidelity of such environments becomes increasingly ambitious. Consequently, much recent research has focused on utilising additional sensory stimuli to create compelling environments [DWS*99, SSM*99, HMG03, TGD04]. Audio cues have been shown to increase participants' sense of presence [PC99b, LVK02] and studies have shown that in environments incorporating audio, subjects perform better in spatial localisation tasks than in the absence of this stimulus [ZF03, TGD04].

Haptic feedback has emerged as another important stimulus to exploit in many VR applications. More recently it has been incorporated in a number of single user and shared demonstrator applications [MBB*02, GHAH03]. Many are simple testbed tasks used to assess the value of collaborative haptic manipulation [BOH97, Hub02, OMJ*03, Rei04, KKT*04]. While they provide interesting results from a 'human factors' point of view, it is difficult to transfer these to building shared applications designed to support haptic collaboration. In particular the impact of network issues of latency (the delay between the transmission of a message and its reception) and jitter (the variation in latency) commonly found on the Internet is not a solved problem.

Of all the interaction modalities employed in multi-sensory environments, supporting haptic collaboration over wide area networks (WANs) is the most challenging due to performance requirements of the tightly coupled nature of the interaction. Therefore, we will focus on potential pitfalls which arise in this case, and how to begin to mitigate them.

A number of experimental studies (detailed later) have shown that haptic feedback is conducive to increasing a sense of presence (a feeling of 'being there') and co-presence (a feeling of 'being together') between multiple participants working towards a common goal. Furthermore, the addition of haptic feedback as a means for input and constrained motion can lead to more user-centric interfaces [KW03].

However simply adding additional sensory cues such as haptic feedback to collaborative virtual environments does not, in itself, lead to compelling or usable applications. It is necessary to consider how the supported sensory stimuli can combine, together with rich behaviour, to reinforce the graphical and behavioural fidelity presented to the user. In training applications, a poor combination could potentially reinforce erroneous behaviour.



(a) Virtual biplane model (b) Real biplane model

Figure 16: *Virtual and real biplane models [AKH01]*

6. Effects of Haptic Feedback on Task-Performance

A number of studies, some of which are detailed in §6.1 and §6.2, have been carried out to assess the effects of haptic feedback on users' ability to perform assembly, tapping and co-ordination tasks. A common limitation with these studies is that the results can be mixed, device or task dependent, and often difficult to directly map benefits to real applications.

6.1. Single-User Case

Some single user studies have shown that the addition of haptic feedback improves task performance [GSW97, HGMR98]. In this section we will detail two in particular that are especially interesting: the first is a virtual prototyping task carried out both in a haptically enabled training environment, and with a real model. The second more abstract study is a landmark molecular docking project.

A single user experimental virtual prototyping task to assess the effect of haptics on a real training task was presented by Adams et al. [AKH01]. Three groups of participants were asked to attempt to build a Lego model of a biplane shown in Figure 16. One of the groups had no training, one trained using a VR simulator without haptics and one trained on the same simulator but with haptic feedback. The authors compared completion times for the task and found a statistically significant change in performance attributable to the level of training received. In particular those subjects that trained in the environment with haptics out performed the others when building the real model. This experiment supports others that have shown skills gained in virtual environments to be transferable [RAB*00, TSRD01].

Early molecular docking research projects at The University of North Carolina, in particular GROPE, demonstrated that haptic display (when used to augment visual rendering) can improve user's perception and understanding of forces and solid entities in virtual environments [BOYBK90]. Disappointingly, they found that typically only a two-fold improvement on task performance over a purely visual display was possible. However, this could be attributable to the limitations of available haptic devices at the time. Even now, current devices are not yet technologically advanced enough to provide both adequate tactile and kinesthetic feedback. Most importantly, however, Brooks et al. reported that the most valuable contribution from the additional haptic display to be in terms of user's increased levels of understanding of the physics of drug docking sites. Although the direct benefits of this may be difficult to quantify, for certain educational tasks this increased awareness would provide substantial benefits.

Now that we have outlined details of two important single user studies which analyse the effects of haptic display on task performance, it is appropriate to extend this to multi-user tasks. Studies performed in this context typically assess two factors: the first being collaborative task performance, and the second being the level of co-presence experienced by the participants. We will first detail a subset of those which analyse the basic core effects on shared task performance before describing a selection of experiments relating to co-presence.

6.2. Multi-User Case

Fitts law [Fit54] target acquisition tasks are increasingly being investigated with reference to haptic feedback [AW00, MGFB01]. One such experiment by Eva-Lotta Sallnäs et al. considers a task to hand over a cube to another user, together with tapping at target points within the context of Fitts law [SZ03]. To simulate plausible hand over rules, the model incorporated spring-like forces enabling both participants to feel forces exerted by each other. The hand over of randomly varying-sized cubes was assessed with and without haptic feedback. Prior to the actual assessed task users were trained to use the application with and without haptic feedback, until they felt confident enough to perform the task. This involved moving blocks down a series of shelves, handing over the block, and tapping specific shelves with it. The success of the users performing the task was measured in terms of time taken to perform error free trials, and the average time taken to hand over the blocks.

No statistically significant difference was found between the haptic and non-haptic hand over completion times. However a difference was found in terms of the average time taken to hand over the cube, and the size of the cube leading Sallnäs et al. to conclude that Fitts law holds for this type of task. They suggest that the lack of tangible benefit from the addition of haptic feedback may be due to the task only providing short haptic events as opposed to continued reinforcement. Another possible explanation given is that the spring-like force used to connect the user to the block provided a 'haptic buffer' effect in the interaction, resulting in the recipient being able to move the cube before the first user had relinquished control. One interesting result however, was that test subjects dropped more cubes in the non-haptic version than the haptically enabled version indicating that haptic reinforcement may be valuable in facilitating temporal spatial co-ordination during the exchange.

Another example of a collaborative haptic tapping task was conducted by Margaret McLaughlin et al. This study was interesting because the task involved touch between participants. The experiment used a PHANTOM and a CyberGrasp device, with PHANTOM users being instructed to tap a model of the CyberGrasp user's hand. A tactile 'Morse code' was used which mapped the number of times a particular digit on the hand was tapped, to a letter in the alphabet. The participant receiving the tactile code recorded the letters they believed they had been sent. Results from this study showed that participants found it difficult to determine the codes sent to them haptically. However in most cases this was due to an overestimate of the number of times a particular digit had been tapped, rather than error determining which digit had been tapped. Interestingly a comparison of the most successful and least successful pairs of participants, revealed a potential but tentative correlation between success rate and feeling of co-presence.

Each of these studies indicate very little real benefit from the addition of haptic feedback to collaborative tapping task performance, and are contradictory to the single user examples described earlier. An interesting question to ask is why this might be the case? A number of possible explanations can be conceived, ranging from the choice of task to the choice of device. However in both single user experiments outlined above, the tasks being performed were continuous, intuitive and easy to relate to real world tasks. It is less clear if this is true for the Fitts law tasks described. Haptic interaction should ideally reinforce the user's perception and graphical interpretation of what is happening in the environment, for sighted participants tasks which rely on haptic feedback to dominate over visual display are unlikely to show significant positive benefits. Furthermore, both participants' perception of the environment, and task must correspond. If this is not the case, then the collaboration is likely to break down. Non-intuitive interaction would most likely exacerbate this.

7. Collaborative Experimental Studies

Specific studies involving users collaborating visually and haptically with continuous haptic feedback augmenting the visual display, are increasingly being conducted. These are especially interesting because they report on the sense of co-presence attributable to the addition of haptic feedback. Many such experiments involve two participants performing a simple carrying task in which each participant can feel forces exerted on the entity being shared.

The following set of experiments described are particularly biased towards assessing co-presence, and are perhaps more like the prototyping and docking tasks in the sense that the interaction and both users perception of the environment is more in tune with their expectations.

7.1. Basdogan et al.'s Wire Ring Experiment

The wire ring experiment by Cagatay Basdogan et al. is an important study on collaborative haptic manipulation [BHSS00]. The experiment itself required two users (one expert and one novice) to jointly steer a ring along a wire (shown in Figure 17), while minimising collisions. Only visual feedback was provided upon collision between the ring and the wire. Haptic feedback was employed purely to facilitate the communication between users manipulating the ring. The experiment was carried out first with only visual feedback, then repeated approximately ten days later with haptic feedback.

Two metrics were employed to assess the success of the task; a performance value and co-presence value (based on presence questionnaires [SW97, SSU00]). Mean and standard deviation values for their performance scores show a considerable improvement in task performance attributable to the addition of haptic feedback. Further, they show that the order in which

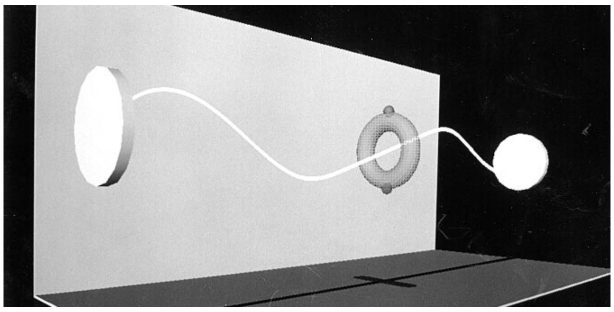


Figure 17: *Basdogan et al.'s wire ring experiment [BHSS00]*

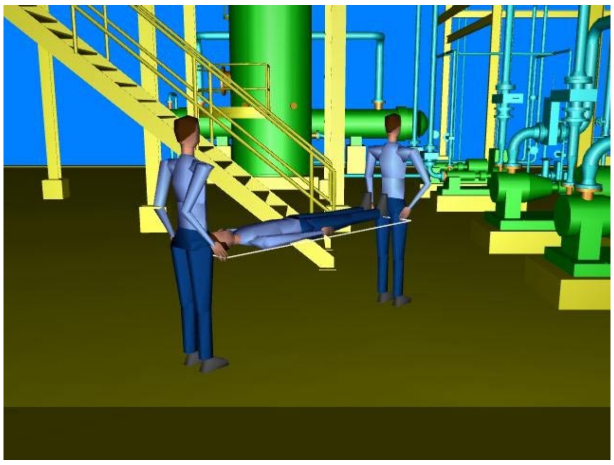


Figure 18: *Hubbold's stretcher carrying experiment [Hub02]*

participants carried out the experiments seems to matter. Groups which first used the visuals only version followed by the visual and haptic display outperformed those which performed the tasks in the reverse order. The initial visual only experience appears to provide a training effect for the augmented haptic environment.

In terms of co-presence, the study reported that in almost all cases participants reported an increased sense of being together in the haptically enabled environment. Further they found that factors such as age, computer usage, and gender differences seem to contribute to the level of co-presence reported. Interestingly they also report that user's perception of the gender of the expert user differed between the haptic and non-haptic task. A significant number of participants believed the expert user they were collaborating with in the haptic environment was male.

7.2. Hubbold's Collaborative Stretcher Carrying

Hubbold's stretcher carrying experiment is unique in the sense that unlike other similar experiments, the task is performed within the broader context of a complex virtual environment [Hub02]. The task itself requires two users to collaboratively carry a stretcher through a model of a process-plant shown in Figure 18. The simulation includes full collision detection and a force field based navigation technique to enable routing the stretcher, and three avatars – two carrying the stretcher, and the patient – through the model. A comparison between participants actually carrying a stretcher loaded with sandbags, and the virtual environment simulation was also undertaken.

Qualitative findings reported from the study included the observation that the virtual environment provides a good predictor for situations in which manoeuvring the stretcher is difficult. This finding was confirmed with the real stretcher carrying task. A second and compelling finding was that the addition of haptic feedback simplified routing the stretcher through the model considerably over the non-haptic version.

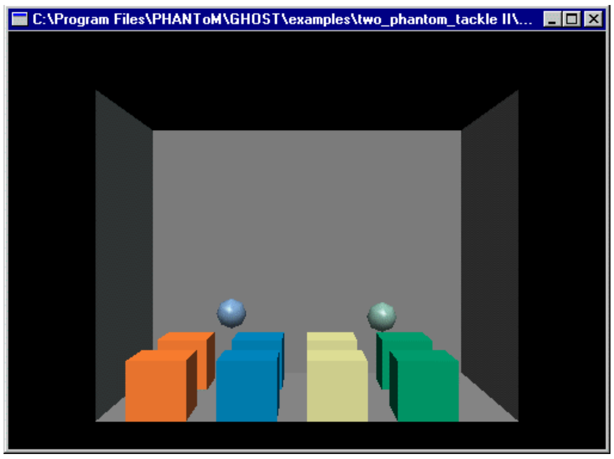

C:\Program Files\PHANToM\GHOST\examples\two_phantom_tackle II\...

Figure 19: *Sallnäs et al.'s cube lifting task [SRGS00]*

7.3. Sallnäs et al.'s Cube Lifting Task

An interesting factor in the analysis of Sallnäs et al.'s cube lifting task [SRGS00] is the use of video taped recordings of participants performing the task together with presence questionnaires allowing a detailed post experimental analysis of task performance. Two out of five tasks were analysed, each requiring two participants (represented by spheres) to jointly lift cubes. The first involved both participants lifting eight cubes (shown in Figure 19) in a specific sequence with the aim to build one cube, without a visual illustration of the target solution. The second task required the participants to collaborate to build two piles of cubes in a specific sequence from the eight cubes in the environment. For this task both participants were provided with a visual illustration of the target solution.

Participants carried out the tasks both with and without haptic feedback, using the haptic device purely for input in the non-haptic case. A two minute training session was employed to familiarise participants with the features of the environment before they undertook the specific tasks. Reported findings from the experiment were somewhat mixed, task performance was shown to improve with the addition of haptic feedback but no significant increase in co-presence was reported. Interestingly however, there appeared to be a statistically significant difference between the participants ability to lift the blocks depending on whether the target solution was supplied, implying (as one would expect) that a prior knowledge of the objective facilitates collaboration.

8. Main Challenges

The benefits offered by haptic interaction in shared virtual environments is not exactly clear cut. This is due to any number of possible factors ranging from choosing the correct metaphors to facilitate human computer interaction to device limitations. However, in the right application context and with appropriate interaction metaphors, a significant benefit can be achieved as evidenced by the GROPE project [BOYBK90]. When developing collaborative haptic applications, it is essential to ask the following questions:

- What types of interaction will the application need to support?
- Will the interaction 'feel' intuitive?
- Will employing haptic feedback greatly benefit my application?

Each of these questions require careful consideration as the answers will impact significantly on the usability and success of the application. A significant challenge is the design of interaction metaphors for haptic communication. For example, does the user manipulate an entities position directly or indirectly? This is very much an ongoing research problem, especially as it is heavily application and device dependent. Further investigation is needed to determine the types of interaction metaphors that users prefer when undertaking classes of haptic tasks. We do not propose to solve the problem here, rather in §9 we present the types of interaction modes possible in shared haptic virtual environments and the application contexts within which these can be employed.

Two more major challenges are briefly outlined in the following subsections: those of shared manipulation of the same entity and distribution over the Internet.

8.1. Shared Manipulation

Two or more participants manipulating the same entity poses a major problem for collaborative haptic environments. There are two significant and related barriers to achieving such closely coupled interaction. The first is to do with the choice of simulation methods, and the second is synchronisation of state information across participants involved in the collaboration.

Robust simulation is required in situations where it is critical that the position computed for the shared entity is constrained and absolute, as is the case for articulated rigid bodies. A solver needs to be employed in order to solve complex constraints on motion, and this inevitably increases the complexity of the simulation. Many of the simple collaborative haptic demonstrators tend to employ fairly basic force models often using spring-like forces to indirectly manipulate shared entities rather than absolute positional constraints. One of the reasons for this choice is that this provides a built in tolerance in the actual position and orientation that the entity may occupy. For many applications this may suffice but for complex environments and critical task training, more sophisticated simulations and force models may be more appropriate.

Related to the issue of simulations is the problem of ensuring that each participant sharing an entity has a consistent view of the environment. The consequences of differing state could be catastrophic as each user's perception of the interaction being performed would differ thus leading to a breakdown in the collaboration. Exchanging state information between participants on a local area network (LAN) is generally fast and reliable, however a wide area network (WAN) such as the Internet brings with it some major challenges detailed later.

8.2. Distribution over the Internet

An interesting point to note regarding all of the collaborative haptic studies described earlier is that none of them operate over a real WAN such as the Internet. In practise there are few studies in which this has been attempted, an example of one such study is the collaborative stretcher carrying example by Manuel Oliveira et al. which illustrates the potential difficulties that arise [OMJ*03]. The task required two participants, one at University College, London and the other at The University of North Carolina to collaboratively carry a stretcher using force feedback devices. In preliminary trials using the Dive [Hag96] system, the authors concluded that the task was not possible due to rapid divergence (caused by network latency) between the haptic and graphical states. Consequently a bespoke application was built to attempt the task, however the authors concluded with a need for a better physical model and strategies to synchronise simulations as well as mitigate the effects of network latency.

Techniques for synchronising state across multiple participants on a WAN are an important consideration for collaborative virtual environments and depend primarily on distribution architecture choices. For behaviour-rich multi-sensory environments these issues become even more crucial as potentially complex state information needs to be kept consistent. In order to better understand how participants may collaborate in haptically enabled virtual environments, it is useful to consider the types of interactions that applications may be required to support. These interactions, many of which may modify the state of the environment, also need to be distributed across multiple participants.

9. Classes of Interaction in Shared Virtual Environments

Broadly, the types of interactions possible in shared virtual environments are either collisions, touch, or more complex actions which have causal relationships with the user or other entities in the environment. In the following subsections we break down this broad categorisation further, in order to detail the types of complex interactions possible.

9.1. User-Entity

User-entity interactions are those which occur between a user and any specific entity in the environment. Figure 20 shows a user (represented by a small shiny black sphere) colliding with a torus. This collision may not provide a haptic response thus it is considered separately to the case of a user touching an entity shown in Figure 21.

Continuously touching and exploring the surface of an entity is an altogether different user-entity interaction relationship as the user's actions and haptic response are closely coupled.

A third user-entity interaction relationship is that where the user moves the entity as shown by a sequence in Figure 22 (muted versions of the user's sphere representation and the torus are historical positions and orientations). Haptically moving a entity may take one of two forms, direct manipulation by selecting it and dragging/dropping it or indirect manipulation by exerting a force on it. Direct manipulation is more akin to lifting and moving objects in the real world however in haptic environments the latter is more commonly employed.

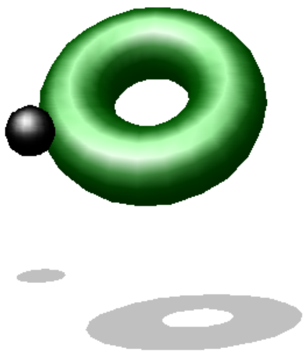


Figure 20: *User colliding with an entity*

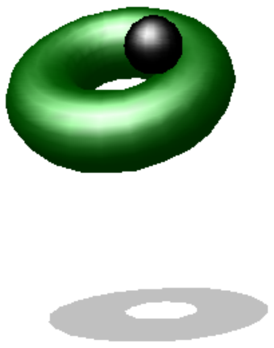


Figure 21: *User touching an entity*

9.2. User-User

In haptic environments, possible user-user interactions could typically take the form where a user collides, or touches another user, as shown in Figure 23. Collisions between users, with or without, haptic response may have consequences such as encouraging the user that has been collided with to move out of the way, or both users to back off. Furthermore, in a haptic environment users do not need to collide to touch, potentially they may interact via indirect forces (illustrated as a curved line between both users).

A more common user-user interaction typically found in virtual environments is the ability to converse as shown in Figure 24. In the past, applications have adopted anything from text based communication to 'Voice Over IP'.

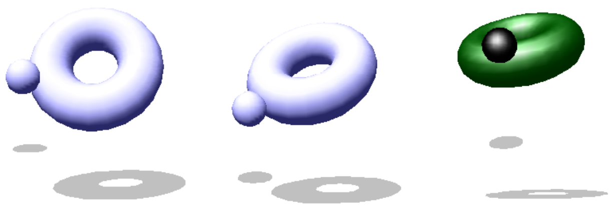


Figure 22: *User moving an entity*

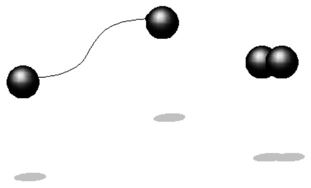


Figure 23: *User touching or colliding with another user*

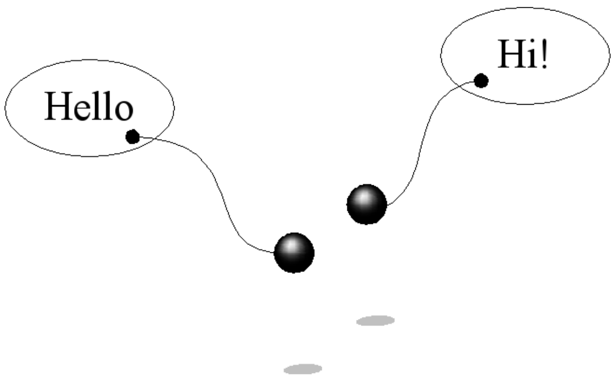


Figure 24: *User talking with another user*

9.3. User-Entity-User

Interactions in the category user-entity-user can be fairly sophisticated. Firstly, a user colliding with an entity can cause it to move and in turn collide with another user as shown in Figure 25. Although this interaction can be seen as two user-entity interactions, it is described here due to the causal relationship of the interaction.

Two or more users may also simultaneously touch the same entity as shown in Figure 26. This situation is not too complex as long as each user does not invade the space of other participants. In the case where users try to fight over a particular region, the simulation becomes complex as each user's representation potentially collides and should provide a proper haptic response in addition to that between the users and the entity.

The interaction scenario shown in Figure 27 where two or more participants manipulate the same entity is possibly the most complex interaction situation possible. To achieve this requires the state to be synchronised between participants in addition to collision detection and haptic response between user representations, and between users and entities. This is the worst case scenario both from a simulation and distribution perspective, especially if the entity being manipulated is complex and changing its state may have a knock-on effect elsewhere.

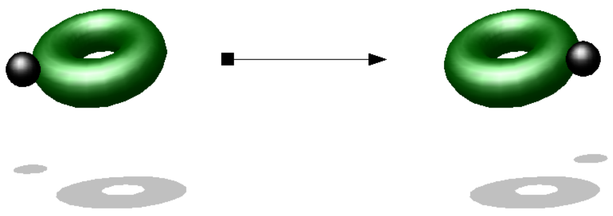


Figure 25: *User colliding with entity which in turn collides with another user*

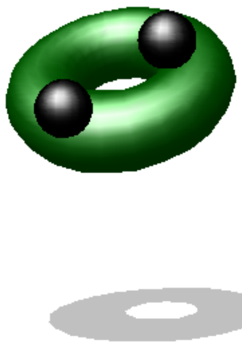


Figure 26: *Two users touching the same entity*

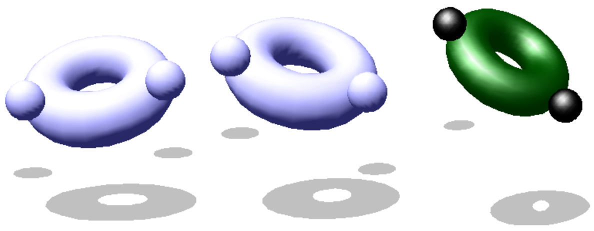


Figure 27: *Two users moving the same entity*

9.4. Entity-Entity

Entity-entity interactions of the type shown in Figure 28 of a torus colliding with a cone are fairly straightforward but may also have consequences depending on the physics of the environment. For example, upon collision the torus may come to a standstill, it and the cone may move off together, or it may cause the cone to move off in a different direction to its continued motion. Haptically however this type of interaction, depending on the complexity of each entity involved, presents a significant collision computation problem within haptic performance demands (algorithms for this are detailed in §17.1).

An entity exerting a force on another entity, as shown by the curved line between the torus and cone in Figure 29 may exhibit complex behaviour depending on the nature of the force between the entities.

Further complex interactions can be achieved through combinations of each of the above described interactions. Consequently, these types of interactions need to be appropriately synchronised between all participants if a meaningful collaboration is to be undertaken.

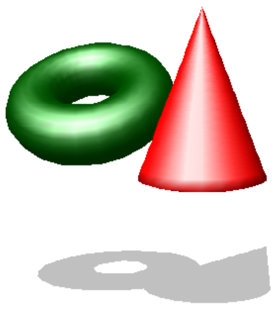


Figure 28: *An entity colliding with another entity*

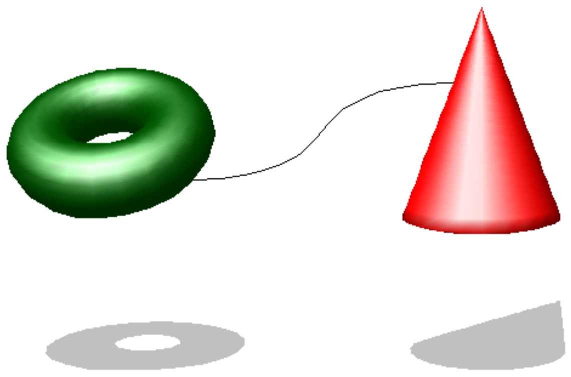


Figure 29: *An entity exerting a force on another entity*

10. Common Shared Interaction Modes

Considering the specific case of shared interaction with the same entity, a number of interaction modes may be employed to manage the collaboration. The mode employed should be dependent on the demands of the application. Shared haptic applications need not support simultaneous manipulation of the same entity unless it is specifically required. In the following subsections we briefly describe three possible shared interaction modes and their practical usefulness in a collaborative application.

10.1. Turn Taking

In a turn taking (or collaborative [BOH97]) interaction mode, each participant collaborates by performing their action in turn. While a user is manipulating the entity, others watch and await their turn. In many prototyping, training and gaming applications this mode of manipulation is both natural and intuitive. The entity being manipulated is commonly locked, using a token, to a specific participant for the duration of their interaction. When the token is relinquished the entity is available to another participant.

10.2. Free-for-All

The free-for-all (or co-operative [BOH97]) interaction mode is tightly coupled with participants simultaneously manipulating an entity. It is commonly found in the real world when people need to move or manoeuvre a bulky heavy object. In virtual environments it has also been employed for similar tasks however, in a training application it may provide an excellent method for participants to interrupt a specific sequence of actions. For example, user *Rob* is demonstrating an assembly maintenance operation to a group of users but *Raffi* has some difficulty understanding the sequence; he may choose to take hold of the entity *Rob* is manipulating, in order to intervene, and pause the sequence.

10.3. Guided Hand

Guided hand is another tightly coupled interaction mode however, it differs from the free for all mode in the sense that there is always one active and one (or more) passive participants. The active or dominant participant may be an expert user training other participants how to perform a complex task. This type of interaction is most likely to be suited to surgical training applications [GHSA05].

11. Distributing Simulations

We have already stated that when distributing simulations over a network, each participant must experience a coherent view of the simulation. This is even more critical in applications in which users may interact with and affect the outcome of the simulation. It would be difficult for multiple participants to undertake a meaningful collaborative task involving interacting with the same rigid or deformable entity, if each person did not see a consistent and plausibly correct simulation of its behaviour. Some state information relating to the simulation (such as positional updates, forces, constraints, changes in size etc.) needs to be sent over a network to other participants. With any network there is some elapsed time (latency) introduced between sending a message and that message being received.

11.1. Injection Problems

Figure 30 shows a number of different simulation scenarios involving two participants each running their own instance of a simulation, over different network conditions, with events from each being *injected* into the others simulation [MGPH05]. The first simulation 30(a) occurs over a low fixed latency connection, events from each participant arrive in time to be taken account of in the simulations and both remain consistent with each other. In the second case 30(b), the simulation occurs over a high fixed latency connection. Each participant's events arrive consistently late, and so the order in which these are incorporated into the simulations differs but in a predictable manner. The third case is more complex and represents a variable latency connection 30(c), notice how the order in which events are introduced into the simulations is much more haphazard and difficult to predict. Finally the most catastrophic case 30(d), in which the latency is variable and events may even be lost, results in very different outcomes for both participants. Depending on the network protocol used, this loss of updates may be impossible to detect. TCP/IP guarantees to deliver updates by transparently re-sending lost updates incurring a significant additional latency. UDP on the other hand attempts to deliver the updates in the order in which they arrive, but offers no notification of lost updates. This can have serious consequences on simulations particularly if the lost update makes a significant state change [MGPH05].

11.2. User's Interaction Information

Depending on the application and the possible tasks that it allows users to perform, a number of different interaction related state updates may need to be distributed. Shirmohammadi et al [SG01] introduce the concept of interaction streams, in conjunction with reliable transfer for key updates, in order to share interaction data to facilitate collaboration. This approach recognises the fact that different applications impose different demands on the reliability and update rate of shared interaction state. Using the concept of interaction streams high priority interactions (such as user-entity-user, or entity-entity) which have a causal effect, could be sent using a different network protocol compared to lower priority ones.

In a simple walk-through, incorporating simulations of flocking birds which the user does not directly affect, only the user's positional updates need to be distributed. Since the purpose of simulation in this case is purely to provide some contextual dynamic behaviour in the environment, each participant's simulations may be computed locally thereby reducing the distribution requirements. If however users are able to direct and change the path taken by the flocking birds, then this change must be distributed to other participants. This can be achieved by either sending positions of all the flocking entities or by sending just the events that caused the change, which is more efficient but may result in subtle differences in each independent simulation. For a purely gaming application, these subtle differences in state may not be critical but this is not always true for all simulations in all applications.

Consider a collaborative CAD application employing deformable object simulations. If two or more participants are required to co-operatively lift and position a deformable component then each user's position and interaction information must be distributed. The question is, how best to distribute the state changes for the shared entity. The answer depends both on the network conditions and the techniques employed to compute the deformations. If an iterative numerical method is employed, stability and convergence [But87] problems can arise between separate simulations, particularly when events are delayed and injected late or lost [BT89]. Consequently participants' simulations will rapidly accumulate errors, and diverge making any collaboration impossible.

11.3. Synchronising State

The problem of divergent simulations is solved by synchronising state information across all participants. This is essentially achieved by maintaining a single locus of control [Pet99] for complex simulations. A single *server* or simulation manager maintains a consistent simulation state, and determines how this is synchronised across all participants. Exactly how this is mapped to distribution policy and architecture, depends on the type of network that the application uses [BT89, Dah99, Fuj99]. There are a variety of different ways in which this can in fact be configured, while still maintaining a logical single locus of control. Real networks are very different and the amount of fixed, variable latency, and update loss differs significantly between LAN and various WAN networks. To better understand how to implement suitable distribution policies and which architecture to adopt, it is necessary to understand the network conditions under which they will be employed.

12. Real Networks

Real wide area network characteristics vary widely but can be typically characterised in terms of two metrics, the amount of latency and jitter. Latency is the delay between a message being transmitted and received and is unavoidable, as no matter how fast networks become they are fundamentally limited by the time taken to transmit data over a given distance through the carrying medium. We normally encounter latency in long distance telephone conversations, when faced with such communications

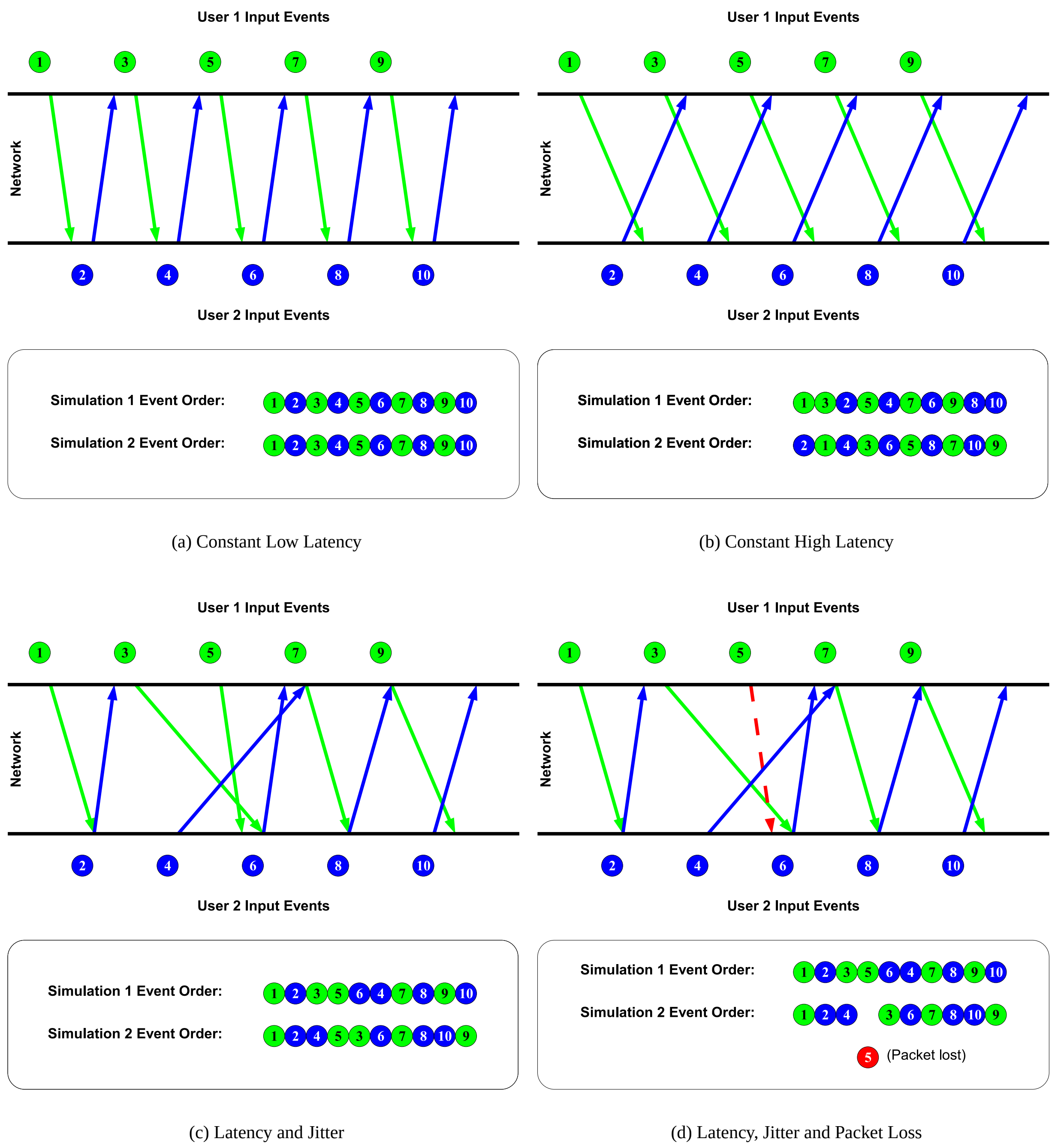

User 1 Input Events
Network
User 2 Input Events
Simulation 1 Event Order: 1 2 3 4 5 6 7 8 9 10
Simulation 2 Event Order: 1 2 3 4 5 6 7 8 9 10
(a) Constant Low Latency
Simulation 1 Event Order: 1 3 2 5 4 7 6 9 8 10
Simulation 2 Event Order: 2 1 4 3 6 5 8 7 10 9
(b) Constant High Latency
Simulation 1 Event Order: 1 2 3 5 6 4 7 8 9 10
Simulation 2 Event Order: 1 2 4 5 3 6 7 8 10 9
(c) Latency and Jitter
Simulation 1 Event Order: 1 2 3 5 6 4 7 8 9 10
Simulation 2 Event Order: 1 2 4 3 6 7 8 10 9
5 (Packet lost)
(d) Latency, Jitter and Packet Loss

Figure 30: *The effect of different network conditions on distributed simulations [MGPH05]*

	Distance	Lost	—— Round-Trip Time (ms) ——				Mean Round-Trip	Typical
Destination	**(km)**	**Packets**	Min.	Max.	Mean	Mean Deviation	**Update Rate (Hz)**	**Hop Count**
Office LAN (Switched Ethernet)	0.01	0	0.36	13.2	0.46	0.047	2170	1
The University of Bristol, UK	230	0	13.9	88.2	14.3	0.34	69.9	14
FCS, Amsterdam, The Netherlands	496	0	20.9	164	27.6	4.54	36.2	18
Labein, Bilbao, Spain (night)	1140	3	62.3	663	68.3	4.59	14.6	17
University of North Carolina, USA	6062	0	104	106	104	0.038	9.62	18
Labein, Bilbao, Spain (afternoon)	1140	68	87	2298	341	136	2.93	17

Table 2: *Results of sending 8000 packets to various destinations from The University of Manchester, UK [MGPH05]*

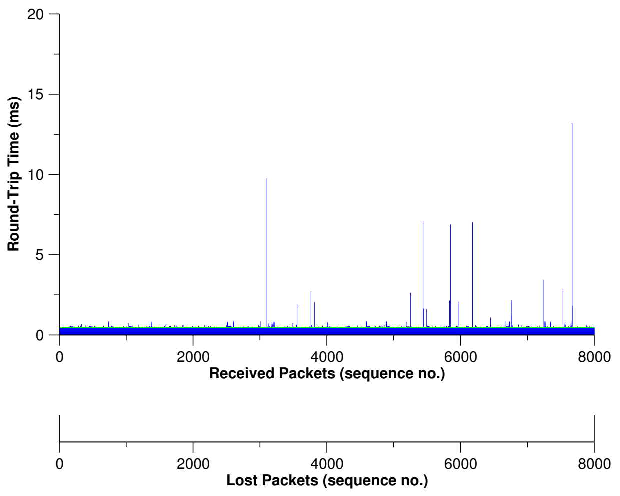


Figure 31: *Measured performance on The University of Manchester local LAN [MGPH05]*

delay, in structured conversations people may adopt a *wait and talk* (turn-taking) strategy or more often curtail the conversation [EM63, RK63]. Latency can increase significantly if a network is heavily loaded as the sender must wait until the network is free of traffic (in the case of Ethernet) before it can send a new message [CDK01]. Jitter; the second metric, is the actual variation in latency, often this can be attributed to packets being lost and needing to be re-sent. While it can be argued that with better network quality of service jitter will cease to be a problem, due to financial cost of improved network infrastructure it is still likely to persist for the foreseeable future.

12.1. Network Latency and Jitter

The aim of any distribution policies and architecture facilitating collaboration, is to present a consistent and coherent view of a shared environment to each participant [MGPH05]. Ultimately the speed of light imposes a finite upper bound on how fast information can be communicated across a network, and in practise real network performance is well below this level. Consequently, regardless of architecture, two remote users cannot share an instantaneous, fully-synchronised view of an environment [Mar02]. Faced with this challenge the majority of collaborative virtual environments adopt one of the two most common network distribution architectures, client-server or peer-to-peer [SZ99]. Both these architectures have their own benefits and shortcomings detailed later.

Table 2 and Figures 31 to 36 illustrate the result of measuring latency and jitter on a number of network routes between The University of Manchester and our research partners. These routes differ in performance by as much as one or two orders of magnitude. Consequently architectural choices appropriate for a LAN are not necessarily going to apply for typical WANs such as those profiled, and likewise applications designed specifically to work over WANs are unlikely to take full advantage of the faster local environment [MGPH05].

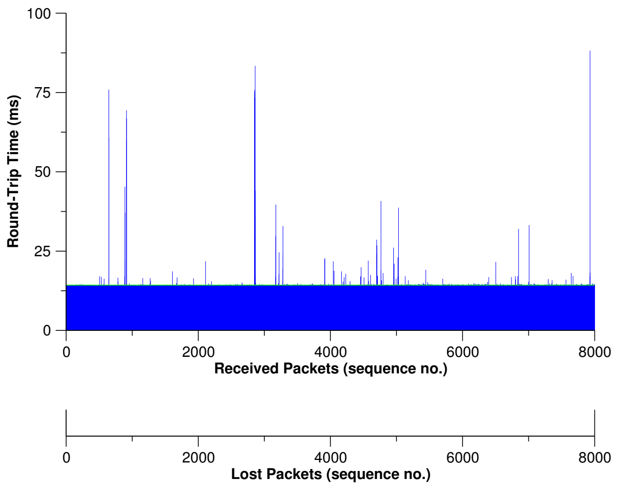


Figure 32: *Measured performance to The University of Bristol, United Kingdom [MGPH05]*

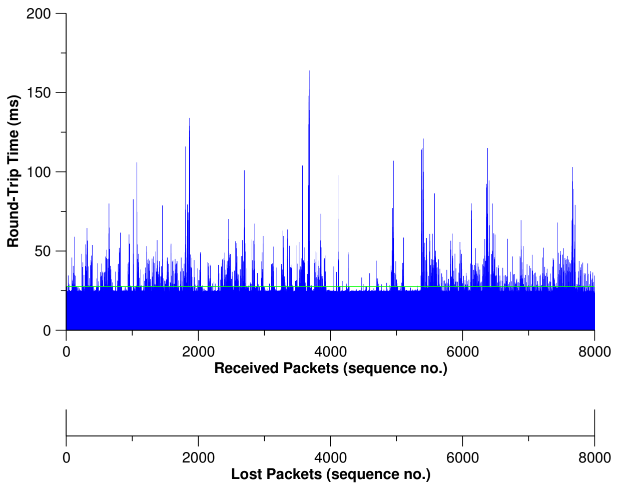


Figure 33: *Measured performance to FCS, Amsterdam, The Netherlands [MGPH05]*

12.2. Impact on Task Performance

A number of studies have examined the effectiveness of collaborative virtual environments and groupware applications. In particular, these have considered factors that affect the extent to which such environments can be successful, including network latency, collaboration strategies, rendering fidelity, and interaction mechanisms.

Participants have been shown to adapt to performing tasks in the face of fixed network latencies up-to 200ms [PK99]. However, the amount of fixed latency tolerable is highly dependent of the task and collaboration strategy adopted. Often in situations where users are able to tolerate high latency, it is attributed to a *move and wait* strategy being adopted. Jitter potentially has a much greater impact on the users' ability to co-ordinate their actions, particularly with regards to predicting the actions others [Gut01].

Changes in participants natural interaction strategies to compensate for the effects of latency and jitter can reinforce erroneous behaviour/responses, demonstrated in studies with teleconferencing applications [RJ99]. In the context of virtual training applications, if a network glitch causes a sudden and surprising discontinuity in motion, the trainee may believe that they caused it by making an error, even though they were performing the correct operation.

For graphical display of 3D environments, one of the most important considerations is maintaining suitably high frame rates.

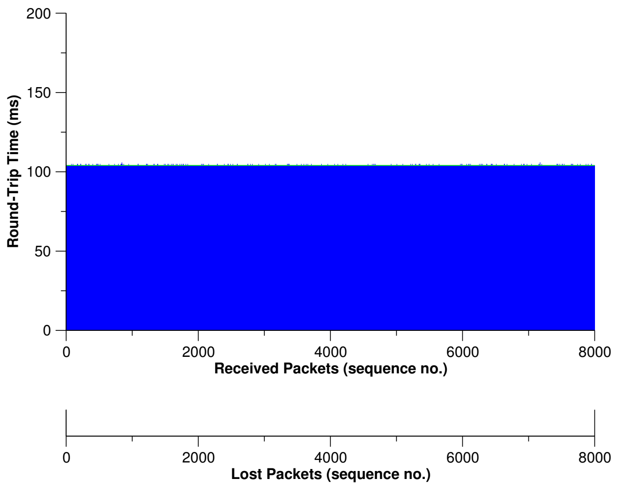


Figure 34: *Measured performance to The University of North Carolina, USA [MGPH05]*

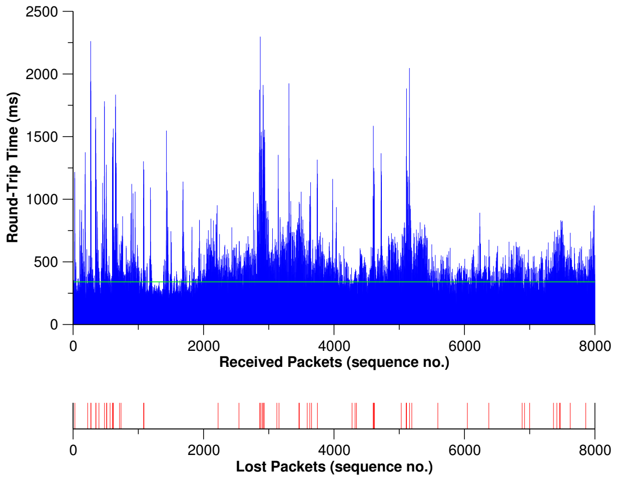


Figure 35: *Measured performance to Labein, Bilbao, Spain (afternoon) [MGPH05]*

This is considered to be approximately 20–30Hz [Bro99]. Below this level motion appears discontinuous, and entity interactions such as collisions may not be correctly rendered due to the high inter-frame latency.

12.3. Architectures for Distributed Systems

The two main architectures adopted for distributed systems are client-server and peer-to-peer, each of these architectures have their strengths and weaknesses and the choice of which to adopt depends on the demands of the application. At the heart of any distribution architecture is a choice regarding where control should reside, and for many applications, this decision is not always intuitive. Therefore, in addition to the basic distribution architectures a number of hybrid approaches have also been proposed. The relative merits of a number of approaches are discussed in the following sections.

12.3.1. Client-Server

The client-server architecture (shown in Figure 37) is one of the most straightforward architectures to use for supporting rich behaviour. It uses a centralised server that runs a simulation of the virtual environment and communicates relevant state changes to all the clients. Managing state across all clients is simple and performed by the server. Clients contain a representation of the

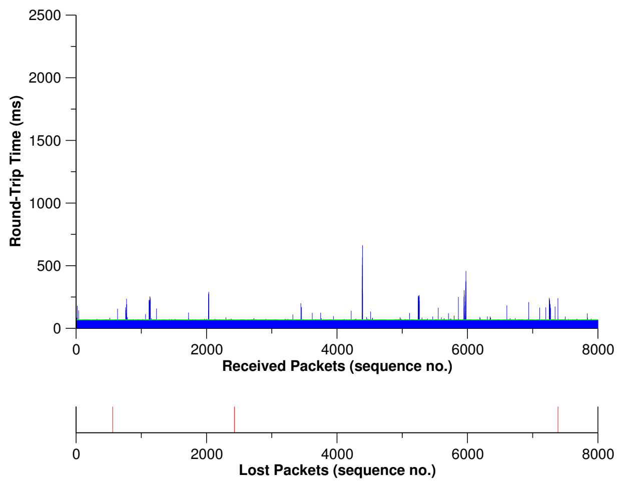


Figure 36: *Measured performance to Labein, Bilbao, Spain (night) [MGPH05]*

application for graphical rendering, but do not perform any simulation activities locally. User interactions are sent directly to the server for processing and clients only update their local representations when instructed to do so [Mar02].

Instead of distributing events which need to be injected into separate simulations, information describing the outcome of a single centralised simulation is communicated to remote clients. This architecture is robust against the effects of jitter and latency, as when update events suddenly experience an increased delay the graphical effect will be no more serious than a temporary disruption of a television programme. The client may briefly display an inconsistent view of the world, but this is quickly corrected when the update does eventually arrive. In addition to the synchronisation benefits, with a single point of contact, application start-up is simplified, and it is straightforward to have the environment persist even when all the clients disconnect [Mar02, MGP*04, MGPH05].

The biggest disadvantage of the client-server approach is that the client's local view of the environment is only updated after a round-trip to the server, resulting in additional input latency for all connected participants. However it is generally accepted that this architecture is the most appropriate to support co-operative tasks in collaborative virtual environments due to the need for consistent state [BOH97].

When latency is too high (as is the case between Manchester and Labein) the graphical update rate requirement of at least 20–30Hz cannot be easily met using a client-server approach. Consequently the user's input is seen to 'lag' and their perceptions of causal relationships begins to break down [Mic63]. The effect this actually has is likely to be task dependent, and experimental studies disagree with regard to the threshold at which task performance is affected. Some reports suggest that participants can tolerate constant latencies of up-to 200ms without a significant degradation in task performance [PK99], while others suggest the figure is much lower [MW93].

Our earlier analysis of the round-trip latencies on our office LAN, and the Internet connections between The Universities of Manchester and Bristol, and between The University of Manchester and FCS-Control Systems in Amsterdam indicate that network conditions are sufficiently fast for a pure client-server architecture to be adequate (assuming no significant latency is added by simulation processing on the server). Occasional peaks of latency higher than 50ms, are likely to cause brief perceptible glitches in continuous movement [MGPH05].

12.3.2. Peer-to-Peer

A peer-to-peer architecture on the other hand has each computer (or peer) directly applying a user's input locally to their own instance of a simulation, while simultaneously communicating these events to other peers as shown in Figure 38. The main benefit of this approach is that it avoids the additional input latency present in the client-server approach, and hence has proved a popular architecture for a number of collaborative systems [MZP*94, BBFG94, GB95, Hag96]. For many applications with low synchronisation requirements (such as multi-player war games, viewing scientific datasets or static models, simple carrying tasks, and primarily social environments) such architectures are successful, particularly over low latency low-jitter connections [MGPH05].

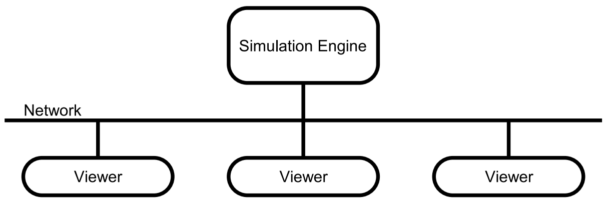


Figure 37: *Client-server architecture [MGP*04]*

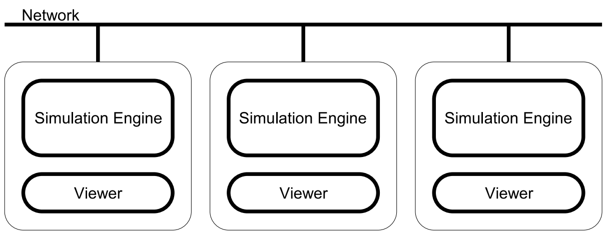


Figure 38: *Peer-to-peer architecture [MGP*04]*

However, in complex behaviour-rich environments synchronisation issues begin to dominate. In the absence of a server to arbitrate, when simulation states computed by each peer diverge catastrophically (due to updates either being discarded, arriving in different orders, or at different times at each peer), there is no obvious way to correct the causal inconsistencies that arise [MGPH05].

Peer-to-peer architectures are most successfully employed on networks with an update delivery rate similar to that shown in Figure 30(a). Events are delivered in the same order to all peers and with a suitable time-stamping and simulation roll-back scheme, updates can be applied at the same time to all simulations. Depending on the network protocol used, however, packet loss can still pose a significant synchronisation problem [MGPH05].

12.3.3. Shades In-Between

Due to the limitations of the two main distribution architectures detailed above, a need exists for hybrid approaches. A third configuration commonly used attempts to combine the low-latency advantages of a peer-to-peer network, with the synchronisation benefits of a client-server architecture. This is achieved through a peer-to-peer system which enforces object-based locking as illustrated in 39 (an architecture occasionally called *token-ring* [BOH97]). This eliminates the problem of out-of-order updates as only one user is able to interact with a given simulation object at any instant in time, unfortunately however it also prevents simultaneous co-operation.

Again, for many applications this architecture will be sufficient, but it imposes strict turn taking which may be unintuitive in certain application contexts. For example in many interactions, such as holding a conversation, social conventions naturally result in turn-taking, but participants still expect to be able to interrupt others [MGPH05].

An improvement to this approach is to provide the ability to identify and lock specific regions of a peer-to-peer network into work groups, in which all related peers share network characteristics similar to those shown in Figure 30(a). An architecture similar to this was suggested by Hespanha et al. [HMS*00].

One problem with architectures such as this, though, is that it is not always clear to the participants who is *allowed* to interact at any particular instant in time. Neither is it clear how best to assign participants to specific work groups. This can result in the situation where users on the periphery of a fast network group are able to collaborate with all other participants except each other, when either one joins the interaction they lock out the other, while the rest of the group appears free to interact at will [MGPH05].

An alternative architecture proposed by Marsh et al. [MGP*04, MGPH05] utilises a number of distributed simulation servers, typically with one per local network as shown in Figure 40. One of these servers is nominated to manage the environment itself, ensuring that logically only one of the simulation engines can be active at any time.

When the system starts up, all of the simulation engines are dormant. Clients connecting profile the network link to each

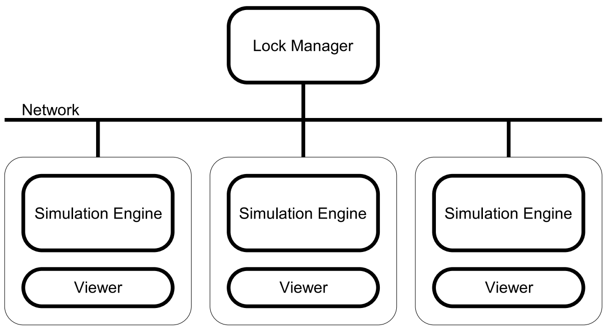


Figure 39: *Enforced turn-taking (locking) architecture [MGP*04]*

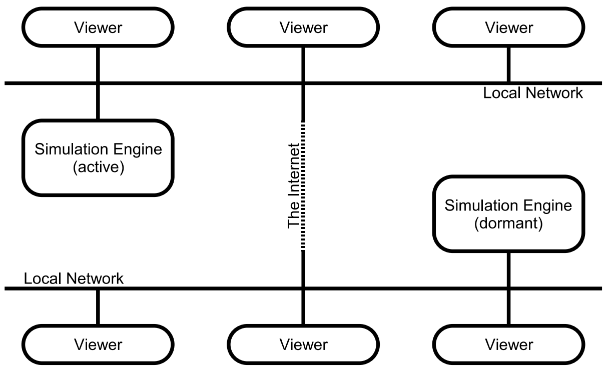


Figure 40: *Roaming server architecture [MGP*04]*

available simulation server, in order to determine their preferred one for subsequent interactions. When a user begins to manipulate an entity, their client requests that its preferred server is activated. If all the simulation engines are dormant, this request is granted, and the client notified. All updates now occur between the client and the active fast *local* simulation server, but in an identical manner to a standard client-server architecture.

If a second user now joins the interaction to either collaborate or co-operate with the first, their client sends a request to activate their preferred server. In this case, the request is declined and the requesting client is informed to use the existing active simulation server. Since both users interact using the same server they are free to interact with the same simulation entities. However the second user may suffer greater latency than the first depending on their geographic location and network connection.

A list of actively interacting users is maintained and if a period of two seconds elapses in which no-one holds any entities, the simulation servers synchronise and enter their dormant state. The system is now ready to activate whichever simulation engine the next user to initiate an interaction requests. If during this two second interval a participant selects an entity then the synchronisation process is cancelled in order to maintain predictable behaviour.

13. Considerations for Behaviourally-Rich Distributed Haptic Environments

We have mentioned the required update rate for graphical rendering, however for haptic rendering a far greater demand exists. It is widely believed that an update rate of at least 1KHz [Bur96] is necessary for correct perception of solid contact; below this rate entities begin to feel ‘spongy’, and if the rate drops too low, instabilities will arise. Given the types of networks profiled, it is a significant challenge to maintain a fast enough update rate for co-operative manipulation of a shared entity. In order to see how this may be supported and the restrictions which apply, it is useful to understand some of the strategies adopted to mitigate the effects of latency and jitter.

13.1. Techniques for Managing Latency and Jitter

A number of techniques have been successfully employed to manage limited latency and jitter for specific applications or tasks. These include using a peer-to-peer approach, time warping [Jef85, MVHE04, Mir00], additional local latency [Mau00], buffering/replay [MPW99], and predictive/dead reckoning [BHM*92, MZP*94, WZ98, WKSKH99] techniques.

While peer-to-peer architectures save on the round-trip requirement of client-server approaches, we have seen that synchronisation issues rapidly begin to cause problems in peer-to-peer systems with heavy simulation synchronisation needs. Thus for behaviour-rich environments, this architecture is only really suitable to employ on fast networks. Many existing peer-to-peer applications work due to the relative simplicity of the behaviour supported.

Computation of costly multi-body simulations is often parallelised. However synchronisation issues present a significant challenge, even to shared memory supercomputers, where communications latency is many orders of magnitude less than that on a LAN. Parallel discrete-event simulation relies on concepts such as virtual time to enabling time-warping, allowing simulations to be rolled back to a consistent state when delayed updates cause conflicts between state. The use of this concept has been suggested for virtual environments [Jef85, MVHE04, Mir00], however its lack of wide-scale adoption stems from the need to maintain a history of simulation state, and the ability to rapidly move forwards through time in order to return to the current time. This either implies significant spare CPU capacity or is likely to have the counter effect of introducing further latency [MGPH05].

The concept of introducing an additional delay to the application of locally generated events *additional local lag* [Mau00, MVHE04] has been suggested in order to resynchronise updates such that they are incorporated as in Figure 30(a). This requires all input events, whether locally generated or remote, to be delayed at each peer to the latency of the slowest network link. While this solution might be reasonable to adopt under the conditions similar to the connection between Manchester and The University of North Carolina (where there is low jitter and constant latency), to mitigate the effects of jitter, all updates must be delayed by the *maximum* latency of the slowest network connection available. For many of the networks shown in Table 2, this threshold is much greater than the mean round-trip time. The Manchester–Amsterdam connection requires that total delay should be consistently 82ms, whereas the mean client-server round-trip would only be 27.6ms. Similarly for the Manchester-Bristol connection a client-server architecture would experience a 14.3ms mean round-trip, but to compensate for even the relatively low level of jitter requires all updates to be slowed to at least 44.1ms when using an additional local lag approach [MGPH05].

The converse solution of delaying upon arrival and buffering updates by the duration of the mean latency plus the mean jitter, which are then interpolated using cubic curves is particularly suitable for applications in which users naturally adopt a turn taking strategy. On slow or jittery network connections, this approach allows passive observers to view smooth, accurate playback of motion sequences [MPW99].

Predictive or dead-reckoning techniques rely on a local representation of entities to continue their motion, based an extrapolation of historical information, in the absence of updates from the server [BHM*92, MZP*94, WZ98, WKSKH99]. If an update arrives which diverges from the dead-reckoned state of an entity, a brief glitch may be perceived. This solution works well for emergent behaviours involving large numbers of autonomous entities, where such brief glitches are relatively unimportant.

Wilson et al [WKSKH99] present a solution to mitigate the effects of latency in networked haptic applications which uses two approaches. The first models the variable-latency communication link as a passive electricity transmission line with varying mass and stiffness (virtual coupling). This creates a system with variable performance but guaranteed stability for the haptic device. The second technique employs haptic dead reckoning, keeping a complete world model on a server and a local simplified model on clients. This allows the clients to haptically display models at the 1KHz update rate, while receiving occasional corrections from the more accurate server model. A similar solution is employed by Marsh et al's 'haptic demon' which maintains a local haptic update rate in the absence of updates from the server [MGPH05].

13.2. Client-Server vs Peer-to-Peer

For the specific networks profiled, it is interesting to see how client-server and peer-to-peer approaches compare. For low latency and low jitter networks like the office LAN both solutions perform very well for applications with high synchronisation demands. The client-server approach is easier to implement from the point of view of provision of synchronisation strategies but ultimately both architectures perform equally well.

In the case of the low latency but high jitter networks, the client-server architecture is more appropriate than peer-to-peer, as the additional cost of employing techniques such as additional local latency to mitigate the effects of jitter is prohibitive.

For high constant-latency networks, the peer-to-peer architecture offers a good solution to mitigating latency by minimising the round-trip time, especially when used in conjunction with predictive techniques for hiding latency. However, for high latency and high jitter networks a hybrid architecture is necessary as neither client-server not peer-to-peer solutions will facilitate meaningful co-operative collaboration. In a turn taking application however, the above mentioned buffering techniques may be employed to present smooth motion to participants using a client-server approach.

13.3. Rich Behaviour vs Haptic Response

Recall that the graphical rendering update rate is required to be in the region of 20–30Hz, while for haptic display it must be 1KHz or better. This fact means that environments employing simulation with purely graphical display can afford more delay than those employing haptic display. Provided the latency introduced by simulation computation is not too high, the required update rate can be met on a LAN [MGPH05]. Although it is much more challenging to mask the latency for haptic rendering over a WAN than it is to do so for the simulation case alone, recent evidence suggests that the amount of haptic latency tolerable is task and device dependent.

Ferrell [Fer66] performed experiments involving two tasks; a contact task and a spring loading task. The study introduced latencies ranging from 0 – 1 second into the two tasks, and participants were trained to use either a continuous interaction or move and wait strategy. In the contact task, the move and wait interaction strategy was preferred while for the spring loading task the opposite was true. The author concluded that a move and wait approach was preferable when haptic response is indicative of a positional error, but less effective than continuous interaction in search tasks. Instabilities arising from latency were observed as bouncing/oscillations during sustained contact. Consequently the primary advantage of haptic feedback, a closely coupled interaction and response loop, is lost through the introduction of latency.

Jay et al. [JH04] show that in target acquisition tasks in which haptic feedback is intermittent and potentially non-essential, latency in the supply of haptic feedback has no perceivable impact. Additionally while a visual latency of 75ms affected task performance, subjects were not aware of the degradation until it reached 90ms. In a subsequent study of a reciprocal tapping task in which haptic feedback was more critical to the task, Jay et al. [JH05] show that while visual latency started to impact upon task performance at around 70ms, haptic latency did not significantly affect it until around 200ms. Subjects seemed to have difficulty perceiving the haptic latency even when it did begin to affect task performance. In both these studies however haptic feedback is related to a discrete event, and so results may differ substantially for continuous manipulation. In a pilot study in which subjects maintained contact with a solid surface, Marsh et al [MGPH05] state that subjects reported surfaces as feeling strange with additional latencies of around 25–30ms. Significant instabilities became apparent at around 40ms. The impact of latency on haptic task performance however is the subject on ongoing research and still requires further study.

The graph shown in Figure 41 shows the latency introduced by collision, geometric constraint [Lab99] and deformable object [Wei02a, Wei02b] computations carried out by the simulation server in a collaborative CAD prototyping application. This performance information was recorded while carrying out a shared assembly task using model of a blood-pressure monitor shown in Figure 42. The two areas of high computational cost are associated with docking complex components, which potentially match a large number of constraint axes. These regions introduce between 10–17ms of computational latency, when this is added to the network latency the total latency becomes too high for stable haptic rendering over a range of the profiled WANs.

14. A Brief Survey of the State of the Art of Collaborative Haptic Environments

The characteristics of the networks profiled, together with the demands of simulation and haptic display, illustrate how challenging a goal shared co-operative haptic manipulation really is. It is nevertheless an active research area, with a number of groups undertaking studies to better understand the problems associated with achieving shared haptic interaction over the Internet. Some of the solutions proposed are discussed in the following sections and are classified broadly according to distribution choices.

14.1. Peer-to-Peer

Manuel Oliveira et al. [OMJ*03] report on lessons learnt, when implementing a co-operative haptic application to manipulate a stretcher using the DIVE virtual reality framework [Hag96] (DIVE provides a peer-to-peer distribution approach with a replicated database [FS98]). Their haptic stretcher carrying task required participants to follow a specified path into a building and position a stretcher onto a target, and was carried out between The University of North Carolina (UNC) and The University College London (UCL). A participant at UNC acted as a confederate during the experiment aiding the UCL subjects to carry out the task.

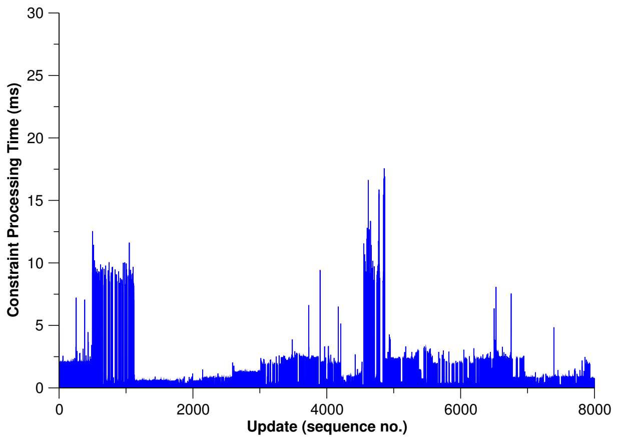


Figure 41: *Latency introduced by the simulation server [MGPH05]*

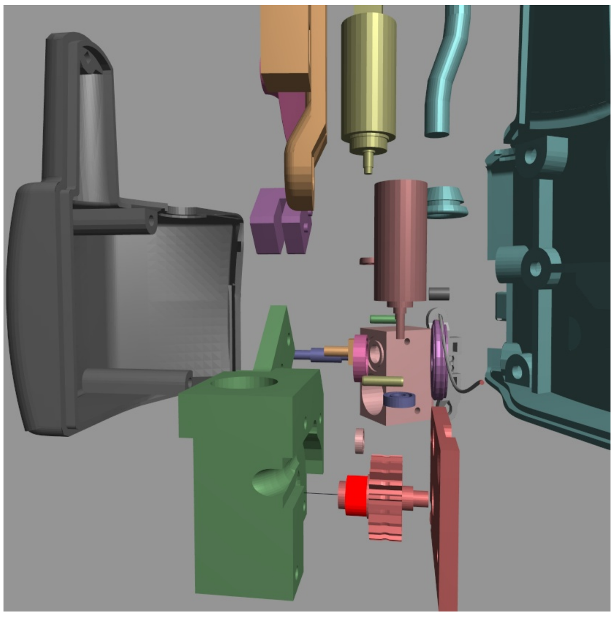


Figure 42: *The blood pressure monitor test case [MGPH05]*

During the experiment the authors found system performance to be especially poor making it impossible to complete the task. They attribute this to the different frequency of updates to parallel data-structures (haptic and graphical). A haptic scene graph was used to locally calculate forces to apply to the haptic device, but both the haptic scene graph and graphical scene graph had to be synchronised for the haptics to reinforce the visuals and visa versa. Since the haptic and graphical update rates are very different, this presented a problem as the event loop in DIVE is tightly coupled to the rendering loop allowing events to only be processed at the rate of graphical rendering. This coupled with network latency resulted in the haptic and graphical scene graphs diverging rapidly. Consequently actual co-operative experiments were performed without haptics, and employed visual and audio feedback however the lack of physical constraints in this case rendered the task less realistic resulting in a loss of presence and co-presence.

Furthermore, the authors experienced other problems with shared manipulation, relating to inconsistencies in user's positions during the simulation caused by a loss or late application of events. In order to address this, they sent several notifications of each event discarding duplicates locally.

The reported result of their study was that joint manipulation of the stretcher using their system posed a number of synchronisation issues. The choice to allow interaction with a local model distributing rotational and translational changes to other remote participants, can be assumed to be synchronised when events are applied in the same order by each client, and generated at a higher resolution than the frequency of the manipulation of the stretcher. If this event delivery characteristic is not met then the divergent states of each instance of the stretcher causes significant jitter of its position. The simulation performed well in tests on the LAN at UCL but when trialled between UCL and UNC the simulation was difficult to synchronise. In order to solve this problem, the authors employed a script to update the local stretcher based on the state of a shared global representation. The stretcher was then aligned locally based on the position/orientation of its handles thus avoiding direct co-operative manipulation.

In an alternative bespoke application two users co-operatively lifted a cube over a height threshold and maintained it there for a pre-determined time. An additional contextual cue (semi-transparent version of the box) was provided to indicate to each participant the state as perceived by the other. The multi-threaded application employed a peer-to-peer architecture communicating simple reaction forces injected locally into the simulation. Updates were transferred via UDP, with the authors concluding that the experiment required sophisticated buffering, a better physical model, synchronisation and smoothing. Further work based on this study assessed the levels of presence and co-presence achieved during the haptic collaboration [KKT∗04].

14.2. Client-Sever

Pietro Buttolo et al. [BOH97] argue that for purely graphical display users adopt a move and wait strategy, to restore hand eye co-ordination. In the case of haptic display, delay in processing information can cause instabilities. Therefore the focus for shared haptic simulation is to reduce delay in processing force information. In order to study architectures for addressing this, the authors investigated three classes of applications:

- haptically enabled environments
- collaborative haptic environments
- co-operative haptic environments

Within the context of these applications, they discuss a scenario involving two participants moving a heavy engine block across a floor, stating that a high quality simulation of this task would be impossible with the typical latencies found on the Internet. Two interaction mechanisms relating to haptic display are were identified, discontinuous *impulsive* (such as kicking a ball) and continuous (such pushing against a wall or lifting an entity). Using these scenarios, the authors suggest how network latency affects the perceived virtual entity's stiffness within the three application contexts identified. They also suggest appropriate distribution choices depending on the synchronisation demands of the application, overall favouring a client-server approach.

In the case of static virtual environments, the authors argue in favour of a client-server architecture if each user requires awareness of the other's actions or peer-to-peer otherwise. For applications employing turn taking, they suggest two alternatives client-server or token-ring. In the client-server case they suggest locking entities using a first come first served decision, while in the token-ring case they discuss ownership of entities according to pre-defined rules. The authors also suggest the use of an *adaptive dynamic bounder* which uses an estimate on the client side of when the next update will arrive, and alters the simulation dynamics according to a server's knowledge of the simulation and round-trip time. They also hypothesise about the possible use of a similar *motion bounder* in which the user's allowable motion is adjusted to compensate for latency.

In the case of applications requiring co-operative manipulation, they argue that a client-server architecture is essential to enforce consistency, suggesting the use of their adaptive dynamic bounder to cope with the round-trip latency.

14.3. Hybrid Peer-to-Peer

Hespanha et al. [HMS∗00] suggest the use of collaborative haptics in a virtual museum application, in which participants can touch and share exhibits which cannot be normally handled in a real museum.

Their proposed distribution architecture partitions the haptic environment into nodes, each consisting of a computer and operator. A dynamic node database is maintained which contains logical IDs, the IP addresses of all the connected nodes, as well as the latency and bandwidth available between them. Simulations are then replicated at each node, with entities locked to specific nodes. State is synchronised periodically through broadcasts. The authors state their solution is not very scalable, particularly when the simulation contains a large number of dynamic entities. They propose partitioning the nodes into local groups sharing a high bandwidth low latency connection to address this problem. Consequently participants on a local group are able to send more frequent updates between each other.

Shen et al [SBNG03] suggest an alternative architecture for collaborative haptic audio visual environments (C-HAVE), based

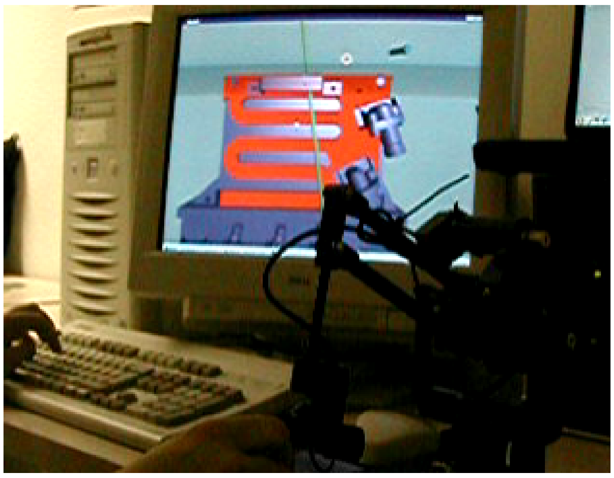


Figure 43: *A collaborative assembly task with haptic feedback*

on the High Level Architecture (HLA) and Run Time Interface (RTI) for parallel and distributed simulation [Dah99]. HLA uses the concept of federations which groups together interacting simulations. Synchronisation between these simulations is managed through the RTI. Rules relating to the ownership of entities in the simulation, define how state is synchronised between connected peers. The C-HAVE system partitions the environment into federates which own regions of interest in which haptic interaction may take place. Each participant subscribes to a region when they enter it and discover the entities available. State information is then shared between subscribed participants. The authors suggest this architecture is potentially suitable for e-commerce applications.

14.4. Hybrid Client-Server

Marsh et al [MGP*04, MGPH05] define and use the roaming-server architecture described earlier in §12.3.3 to manage haptic and non-haptic collaborative CAD prototyping over high latency high jitter networks. Figure 43 shows the Divipro application being used to perform an assembly sequence with haptic feedback through a Sensable PHANTOM device. Smoothing techniques, within the context of the Deva framework, to mitigate the effects of latency were illustrated using the application [MGP*04] and tested between Manchester and Labein. In subsequent work, the use of a haptic demon [MGPH05] to enable collaborative and co-operative hapic manipulation is described. The Divipro application provided the best low-latency haptic experience for collaborative interaction tasks, while still allowing co-operative manipulation. However, depending on network conditions, co-operative haptic manipulation was stable but not ideal for participants remote to the roamed server.

Ongoing research into co-operative haptic manipulation, based on Hubbold's stretcher carrying task [Hub02] is currently being conducted at Manchester. A new implementation of the stretcher task, shown in Figure 44, borrows the concepts of the roaming server and haptic demon from Divipro in the design of distribution policies. The application also uses a more complex simulation of motion, and consequently has a high synchronisation demand. It is currently being tested over a range of different network connections.

14.5. Commercial Solutions

TiDeC a commercial solution, marketed by Handshake VR, to mitigate the effects of latency uses predictive methods to compensate for network delay. The system appears to compensate for delay by profiling the network and predicting the typical delay between each connected participant [AZWS04], once this has been established predictive filtering methods are employed to compensate for the latency [AB95]. Being a commercial solution, little public technical information is available, however the company suggest their solution can compensate for latencies as great as 500ms. A number of pilot studies of the effects of latency on task performance have employed the system, to evaluate whether time-delay compensation can facilitate meaningful collaboration [WTR*03, AZWS04]. Completion times were measured for simple target acquisition tasks, and the authors report that increased latency in visual and haptic feedback adversely affects task completion.

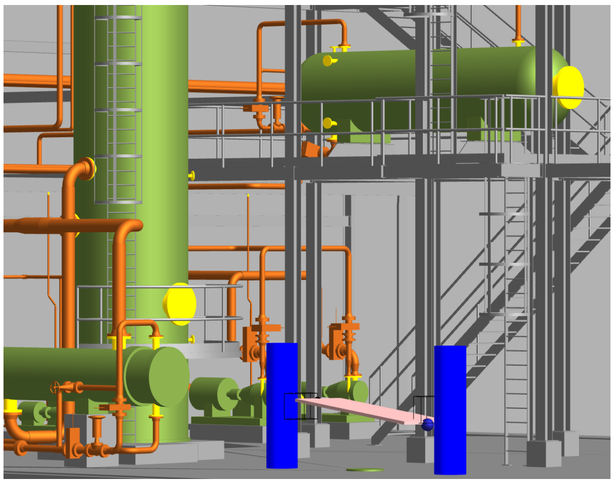


Figure 44: *A co-operative stretcher carrying task*

15. Putting it All Together

It is clear that both performance and network limitations impose restrictions on the complexity of highly interactive shared virtual environments. Furthermore any single distribution architecture is unlikely to support the demands of every collaborative application possible. We have argued that hybrid architectures can be more flexibly exploited to facilitate haptic interaction in behaviour-rich environments. However, there is still no clear general purpose solution to the perennial problem of high latency and high jitter connections. Consequently it is important to determine the type of interaction and the methods to mediate that, within the application context to best decide distribution strategy. In these notes we have highlighted potential problems that need consideration when choosing a distribution policy, and surveyed a number of solutions within the context of shared haptic interaction. Preferred strategies for interaction in shared virtual environments are application and task dependent, and for many applications collaborative (turn-taking) will suffice. While this does simplify state synchronisation, additional cues to mediate transfer of control in the face of latency are essential to facilitate useful collaboration. In shared applications which require co-operative haptic manipulation, hybrid distribution choices must be coupled with techniques to maintain stable haptic interaction. One of the most promising techniques to achieve complex interaction in high quality, behaviour-rich shared virtual environments is to use selective methods exploiting human perception. These methods have been explored in isolation but rarely combined to implement perceptually driven behaviour-rich environments.

15.1. Perceptually Based Distribution

Recall that Deva's [PCMW00] object/subject model for distributing rich behaviour decouples the real state of entities from the perceptual representation. The system's authors recognised that perception plays an important role in simplifying distribution strategies to facilitate shared interaction [Pet99,Mar02]. Central to the Deva approach is Immanuel Kant's philosophy that while we believe our experience of the world to fully objective and un-mediated, it is straightforward to show that our perception of it can easily break down. For example optical illusions such as mirages present an immediate example of our perception failing to comprehend the world around us [Pet99, Mar02].

Kant describes a reality model in which a distinction is drawn between 'things themselves' (he calls the noumenal reality), and 'things I perceive' (which he calls the phenomenal reality). In the context of a virtual environment this model maps to the fact that entities in the environment possess state, and fundamental behaviour, that define their semantic roles. User's in the environment experience this behaviour, by observing the changes in an entity's directly perceivable properties. Therefore, if an entity stores attribute internally as part of its simulated behaviour, but does not exhibit it in some manner it is of no relevance to the user and does not require distributing [Pet99, Mar02].

An early version of Deva simulated the objective reality on a parallel cluster known collectively as the Deva Server with rendering performed by dedicated clients called visualisers. These presented a view of the world and were treated as an extension of Kant's human perceptual pipeline. Only changes in attributes that can ultimately be perceived (by affecting the 3D properties of an entity) were sent across the network. Perceptual filters were used to avoid sending unnecessary updates where the effect would be so small that it would be imperceptible. However, although the system was used to demonstrate the advantages of decoupling the objective and subjective realities for distribution, in practise strictly enforcing distribution of purely perceptible state information was found to be insufficient for maintaining perceptual coherency [Pet99, Mar02]. Consequently further versions of the system relaxed this restriction, while still employing the object/subject model for intelligent local behaviour simulation managed by the server's view of state. This still draws the distinction between the true state, and the perceived state of entities in the environment while allowing more flexible state information exchange [Mar02].

Perceptually guided distribution choices offer the advantage of being able to reduce the state transfer demands of traditionally synchronisation intensive simulations. Approaches based on regions of interest [BBFG94, GB95, GFPB02], and visual attention models to partition the distribution problem according to human visual perception have been successfully exploited [BWH03].

15.2. Graphical, Behavioural and Haptic Correspondence

Systems incorporating high quality graphics, complex behavioural simulations, and haptic feedback often need to maintain different data relating to each. In shared virtual environments this graphical, behavioural and haptic correspondence must additionally be shared between all connected participants. Potentially this can require a substantial amount of state information to be shared.

From a rendering point of view, data describing geometry and material reflectance characteristics must be maintained. Simulations also require models to represent their behavioural responses, and these may in fact be very different to the rendered geometry. For example an avatar may be rendered using surface geometry, while the motion of its body may be simulated using a system of interconnected links and joints. Further, the simulation model itself may be a simplified version of a realistic full model of the human body's bones and joints. Similarly the required update rate for haptic rendering imposes a restriction on the complexity of models that can be displayed. Only a limited number of collision and response calculations can be performed within this tightly constrained update requirement. For this reason, it is necessary to simplify the geometry for haptic rendering. A number of methods have been employed to reduce the complexity of the problem space for haptic rendering, including partitioning environments and caching geometry in the region of interest [GHL05], and sensation preserving perceptual level-of-detail [OL03b].

The latter technique exploits the fact that in many cases, graphical feedback plays a dominant role in user perception of the environment. Consequently as long as haptic feedback supports and re-enforces user expectations from the graphical display, the correspondence between underlying models need not match exactly. By maintaining intact key surface features while simplifying geometry using mesh decimation techniques, it is possible to selectively maintain enough perceptually important features for correct haptic perception. Details of these methods are provided later in §18.

15.3. Providing Richer Haptic Response

To provide richer haptic response in single user and shared virtual environments, it is not only important to maintain a perceptually acceptable mapping between models for graphical and haptic displays, but also use suitable force models to support correct perception of surface detail, entity-entity collisions, and motion during interactive manipulation tasks [OJSL04]. In the following sections the state-of-the-art methods supporting perceptually correct haptic interaction is detailed.

Acknowledgements

We would like to gratefully acknowledge the contributions of Dr. Steve Pettifer, Dr. James Marsh, Prof. Roger Hubbold and Ms. Caroline Jay without whom these course notes would not have been possible. Dr. James Marsh in particular, has provided much of the content on distributed systems architectures, and the Deva System. We would also like to thank our partners, in particular Teresa Gutierrez at Labein; Holger Weiss at DLR; Gunter Kadegge at KL Technik; Ernesto Arenaza at Sener; Bas Ruiter at FCS Control Systems; and Matt Aranha at The University of Bristol for invaluable help with profiling typical networks.

16. Fundamentals of Haptic Rendering

For a long time, human beings have dreamt 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, to *move* them, and to *touch* them. It has been shown that the ability to touch virtual objects increases the sense of presence in virtual environments [Ins01]. Haptic rendering offers important applicability in engineering and medical training tasks.

In §16, §17, §18, and §19 we describe techniques for incorporating haptic rendering to computer graphics applications, focusing on the problem of *six-degree-of-freedom (6-DoF) haptic rendering*, or haptic rendering of the interaction between an object manipulated by the user and other objects in the environment. In this section we introduce the concept of haptic rendering, related fundamental research in psychophysics and control theory, and basic techniques for 3-DoF haptic rendering. In §17, we survey existing approaches to 6-DoF haptic rendering, with an emphasis on the related problems of collision detection and rigid body simulation. The remaining two sections address psychophysics issues in haptic perception, and they describe how perceptual observations have driven the design of efficient 6-DoF haptic rendering algorithms. Specifically, in §18 we describe a multiresolution collision detection algorithm based on the sensation-preserving simplification of complex geometry, and in §19 we describe techniques for perceptually-driven haptic texture rendering.

16.1. Introduction

We start by defining some terminology, discussing the evolution of the research in haptic rendering, and introducing practical applications.

16.1.1. 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. Other haptic rendering algorithms deal 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. Figure 45 shows an example of a user experiencing 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.

16.1.2. From Telerobotics to 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. 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

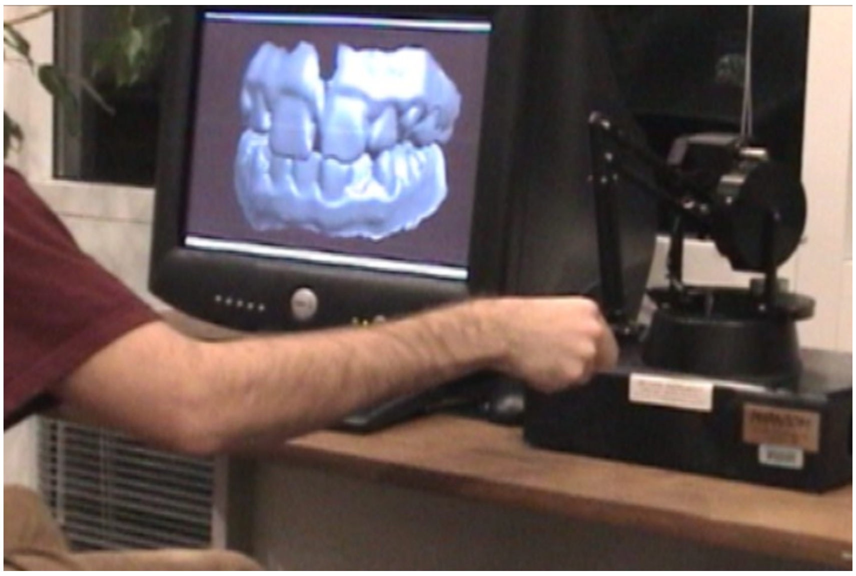


Figure 45: ***Example of Haptic Rendering.*** A person manipulates a virtual jaw using a haptic device (shown on the right of the image), and the interaction between jaws is displayed both visually and haptically.

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* [MOS*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 commercialised 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 we have summarised here. [Bur96] gives a general survey of the field of haptics and [MHS02] discuss current research topics.

16.1.3. Haptic Rendering for Virtual Manipulation

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 behaviours of real and virtual objects. In the real world, solid objects cannot inter-penetrate. 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].

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 generalised to the case of synthetic cutaneous feedback of the interaction with virtual objects. 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.

To summarise, 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.

16.2. The Challenges

Haptic rendering is in essence an interactive activity, and its realisation is mostly handicapped by two conflicting challenges: high required update rates and the computational cost. In this section we outline the computational pipeline of haptic rendering, and we discuss associated challenges.

16.2.1. Haptic Rendering Pipeline

Haptic rendering comprises two main tasks. One of them is the computation of the position and/or orientation of the virtual probe grasped by the user. The other one is the computation of contact force and/or torque that are fed back to the user. The existing methods for 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/or orientation of the haptic device are applied directly to the grasped probe. Collision detection is performed between the grasped probe and the virtual objects, and collision response is applied to the grasped probe as a function of object separation or penetration depth. The resulting contact force and/or torque are directly fed back to the user.

In *virtual coupling* methods [CC97, Ber99, MPT99, RK00, WM03, OL05], the position and/or orientation of the haptic device are set as goals for the grasped probe, and a virtual viscoelastic coupling [CSB95] produces a force that attracts the grasped probe to its goals. Collision detection and response are performed between the grasped probe and the virtual objects. The coupling force and/or torque are combined with the collision response in order to compute the position and/or orientation of the grasped probe. The same coupling force and/or torque are fed back to the user.

In §17, we describe the different existing methods for 6-DoF haptic rendering in more detail, and we discuss their advantages and disadvantages. Also, as explained in more detail in §16.4, 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.

16.2.2. Force Update Rate

The ultimate goal of haptic rendering is to provide force 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 probe touches rigid virtual 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 probe should respond quickly to the motion of the user. Similarly, when the grasped probe 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 behaviour 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.

A more detailed description of the stability issues involved in the synthesis of force feedback, and a description of related work, are given in §16.4. Although here we have focused on impedance-type haptic devices, similar conclusions can be drawn for admittance-type devices (See [AH98a] and §16.4).

16.2.3. Contact Determination

The computation of non-penetrating rigid-body dynamics of the grasped probe 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].

For two interacting virtual objects, 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 §17.1, we 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 force feedback 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.

16.3. 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 behaviour of human touch have been studied extensively in the field of psychology. The topics analysed by researchers include characterisation 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 analyse 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 haptic rendering. In this section we summarise existing research on perception of surface features and perception of roughness, and then we discuss issues associated with the interaction of visual and haptic modalities.

16.3.1. 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. 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.

Okamura and Cutkosky [OC99, OC01] analysed feature detection in robotic exploration, which can be regarded as a case of object-object interaction. They characterised 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 summarise, the studies by Klatzky and Lederman [KL95] and Okamura and Cutkosky [OC99, OC01] lead to the observation 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. This observation has driven the design of multi-resolution contact determination algorithms for haptic rendering [OL03b], described in §18.

16.3.2. 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 characterisation 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]. 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 analyse 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: inter-object 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]. The influence of these factors has been addressed in the design of haptic texture rendering algorithms [OJSL04], described in §19.

The experiments conducted by Klatzky and Lederman to characterise 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 analysed by plotting subjective roughness values vs. dot inter-spacing in logarithmic graphs.

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 characterised 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. 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. 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. 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.

16.3.3. 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 analysing 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 numbers of visually homogeneous objects.

16.4. 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 §16.2.2, we have briefly discussed the importance of stability for haptic rendering, and we have introduced the effect of the force update rate on stability. In this section we review and discuss in more detail existing work in control theory related to stability analysis of haptic rendering.

16.4.1. 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 §16.2.1, 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. Following the unifying framework presented by Adams and Hannaford [AH98a], contact determination and force computation algorithms are often independent of the control strategy.

16.4.2. 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 modelled easily using stiffness and viscosity parameters.

Ouh-Young [OY90] created a discrete model of the Argonne ARM and the human arm and analysed the influence of force update rate on the stability and responsiveness of the system. Minsky, Brooks, et al. [MOS*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 analysed the influence of position sensors and quantisation, and concluded that sensor resolution must be maximised 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.

16.4.3. Passivity and Virtual Coupling

A subsystem is *passive* if it does not add energy to the global system. Passivity is a powerful tool for analysing 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 behaviour 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 behaviour [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 behaviour of the force-feedback system. This active behaviour 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 modelled 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:

$$b > \frac{KT}{2} + B. \tag{1}$$

Stability of Non-Linear Virtual Environments

After deriving stability conditions for rendering virtual walls modelled 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 analysed 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 analysing 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. [MCF99] 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.

16.4.4. Multirate Approximation Techniques

Multi-rate 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 multi-rate approach has been followed by many authors for haptic interaction with deformable models [AH98b, CT00, DAK04]. Ellis et al. [ESJ97] produce higher-quality rendering by up-sampling directly the output force values.

16.5. Three-Degree-of-Freedom Haptic Rendering

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 summarised 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, BSLM01, FOL02, AWD*04], volume visualisation [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.

This section briefly describes some of the most popular 3-DoF haptic rendering techniques, haptic texture rendering algorithms, and special methods for rendering large data sets. The section concludes with a discussion on what applications are efficiently solved with 3-DoF haptic rendering, and what applications require 6-DoF interaction.

16.5.1. Constraint-Based Methods

As mentioned earlier in this section, 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], the closest point may jump arbitrarily along the surface. As a result, closest-point methods lead to force discontinuities and possible "pop-through" problems, in which the contact point jumps between opposing sides of the object.

Zilles and Salisbury proposed the *god-object* method to solve the discontinuities induced by closest-point methods. Knowing the position of the haptic probe in the current frame and the previous frame, they identified the set of surfaces that constrained the inter-frame motion of the haptic probe. Given this set of active constraints, they computed the position of the contact point as a constrained optimisation problem, using Lagrange multipliers. In particular, they defined a cost function based on the distance between the probe point and the contact point, and they added penalty terms due to the constraint surfaces, weighted by Lagrange multipliers. Then, they minimized the cost function and solved for the contact point. In practise, with Zilles and Salisbury's formulation the contact point may be constrained by one to three surfaces (i.e. plane constraint, concave edge constraint, or concave vertex constraint). Finally, they computed feedback forces based on the distance between the probe point and the contact point.

Ruspini et al. [RKK97] followed a similar approach to Zilles and Salisbury, which they called *virtual proxy*. They modelled the contact point as a sphere of small radius and solved the optimisation problem in the configuration space. As opposed to the *god object* method, they also formulated the problem of identifying the set of active constraints from the local neighborhood in configuration space. At each frame, Ruspini et al. set as goal the position of the probe point. They identified possible constraint surfaces using the ray between the old position of the *virtual proxy* and the goal position. Then, they solved a quadratic optimisation problem and found a subgoal position. They repeated this process until the subgoal position was fully constrained. Ruspini et al. realised that the quadratic optimisation problem for a spherical proxy was a convex one, and could be solved efficiently in the dual space defined by the normals of the constraint planes. 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 behaviour to the contact point).

16.5.2. Scalable Collision Detection for 3-DoF Haptic Display

When haptically rendering large environments or data sets, the rendering algorithms face two main problems: memory limitations, and an excessive cost of collision detection routines. In 3-DoF haptic rendering, however, the high spatial coherence of the collision queries can be exploited to design effective rendering algorithms. A number of researchers have followed this research direction. For example, Glencross et al. [GHL05] have proposed a caching technique for haptic rendering of up to hundreds of thousands of distinct objects in the same scene. Their haptic cache maintains only objects in the locality of the haptic probe.

Others have addressed the problem of haptically rendering large terrain data sets. The methods proposed by Walker and Salisbury [WS03] and Potter et al. [PJC04] both rely on the assumption that the geometric data to be rendered is a height field. The proxy graph algorithm presented by Walker and Salisbury [WS03] restricts the position of the proxy contact point to remain on edges and vertices of the rendered geometric surface. With this limitation, they achieve significant speed-ups in the collision detection process, and they are able to sweep over a large number of surface triangles on one haptic frame. The rendering algorithm presented by Potter et al. [PJC04] describes the height field data as a bilinear patch instead of a triangulated surface. Ray surface intersection tests, and closest-point search operations are optimised in the situation where the surface can be simply described by its height value.

Earlier, Gregory et al. [GLGT99] presented *H-Collide*, a fast collision detection algorithm for 3-DoF haptic rendering. As described in the previous section, constraint-based 3-DoF haptic rendering methods rely on ray-triangle intersections for identifying active constraint planes. *H-Collide* constructs a hybrid hierarchical representation of the scene's geometry as a preprocessing step. First, it partitions the scene geometry based on a uniform grid. Then, for each cell in the grid, it computes an oriented-bounding-box hierarchy (OBB-Tree) of the triangles that lie in that cell. And, last, it stores pointers to the OBB-Trees using a hashing technique. The ray-triangle intersection tests are performed in two steps. In the first step, the ray is hashed against the grid cells. This returns a set of OBB-Trees. In the second step, the ray is intersected against the OBBs and, if necessary, against triangles stored in the leaves of the OBB-Trees. The high spatial and temporal coherence of 3-DoF haptic rendering makes spatial partitioning a very convenient culling approach, as the ray may not be tested for intersection against surface areas that are far from the region that the haptic probe is currently exploring.

16.5.3. Haptic Texture Rendering

Three-DoF haptic rendering algorithms have been extended to account for sub-feature geometric detail that is not directly encoded in the geometric primitives, in a way similar to the texture mapping technique broadly employed in computer graphics. Indeed, haptic rendering of textures was one of the first tackled problems in the field of computational haptics by Minsky et al. [MOS*90]. This section begins with a description of Minsky's pioneering algorithm for rendering textures on the plane [Min95]. Then it discusses rendering of textures on 3D surfaces, covering offset-based methods and probabilistic methods.

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 synthesises a force **F** in 2D based on the gradient of the texture height field *h* at the location of the probe:

$$\mathbf{F} = -k\nabla h. \tag{2}$$

Minsky also analysed 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].

Methods Based on Surface Offsets

High-resolution surface geometry can be represented by a parameterised coarse mesh along with texture images storing detailed offset 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 offset 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 offset 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.

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] synthesised 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. [PDJ*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 synthesising high-frequency effects in 6-DoF haptic rendering.

16.5.4. Three-DoF Vs. Six-DoF Haptic Rendering

As addressed earlier, for many applications the interaction between the subject and the virtual objects is sufficiently captured by a point-surface contact model, and 3-DoF haptic display suffices. 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 modelled 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 insertion of a peg in a hole, 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 modelled 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 modelled in practise by ray-surface contact [BHS97]. In particular, several surgical procedures are performed with 4-DoF tools (e.g., laparoscopy), and this reduced number of DoFs has been exploited in training simulators with haptic feedback [CTS02]. Nevertheless, these approximations are valid only in a limited number of situations and cannot capture full 6-DoF object manipulation.

§17, §18, and §19 cover the problem of 6-DoF haptic rendering, describing the most popular techniques, and focusing on perceptually-driven rendering methods.

17. Six-Degree-of-Freedom Haptic Rendering

From the computational point of view, 6-DoF haptic rendering is tightly coupled to the fields of collision detection and rigid body simulation. Therefore, this sections starts with a description of state-of-the-art techniques in those two fields. The section continues with a description of the existing approaches to the problem of 6-DoF haptic rendering. §18 and §19 describe in detail perceptually-driven collision detection algorithms and force models. The perceptual concepts that drive the design of those algorithms and models can be applied as well to other rendering techniques described later in this section.

17.1. 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 we focus on collision detection for polyhedral models. The vast majority of the algorithms used in practise 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, we first describe the problems of interference detection and computation of separation distance between polyhedra, with an emphasis on algorithms specialised for convex polyhedra. Then we survey the use of hierarchical techniques, algorithms for the computation of penetration depth, and multi-resolution collision detection. We put special emphasis on two algorithms, SWIFT++ [EL01] and DEEP [KLM02a], as they serve as the implementation basis for the perceptually-driven multiresolution collision detection algorithm described later in §18. We 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].

17.1.1. Proximity Queries Between Convex Polyhedra

The property of convexity has been exploited in algorithms with sub-linear 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 optimisation algorithm (known as GJK) for computing the separation distance between convex polyhedra, with linear-time performance in practise. Cameron [Cam97] modified the GJK algorithm to exploit motion coherence in the initialisation of the convex optimisation at every frame for dynamic problems, achieving nearly constant running-time in practise.

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 [Mir98a] 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 hierarchical 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].

17.1.2. 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 $\mathrm{O}(1)$ and that the size of the leaf clusters is also $\mathrm{O}(1)$. 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 generalised 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.

SWIFT++ - Fast Collision Detection Using Voronoi Marching

The SWIFT++ collision detection algorithm by Ehmann and Lin [EL01] is based on BVHs of convex hulls. Ehmann and Lin defined both the process to create the BVHs and the collision queries using convex polyhedra. The algorithm imposes some restrictions on the input models, as they must be orientable 2-manifold polyhedral surfaces, but it offers equally good or better performance than other collision detection algorithms for a variety of proximity queries. Specifically, its performance is as good as that of algorithms based on BVHs of OBBs for intersection queries, but it offers the capability to return additional collision information (i.e. distances, closest points, contact normals), which is very useful for collision response. The reason behind this additional capability is that the collision queries are performed between convex surface patches of the original objects, rather than using bounding volumes with an offset (e.g. OBBs, AABBS, k-DOPs, or bounding spheres).

As a preprocessing, a convex surface decomposition must be performed on the input objects. Ehmann and Lin extended the convex surface decomposition procedure of Chazelle et al. [CDST97], by setting additional constraints such that the convex volume that encloses a convex patch does not protrude the surface of the object. The convex volumes that enclose the initial patches constitute the leaves of the BVH. Then the hierarchy is constructed by pairwise merging convex patches and constructing the convex hulls of the unions.

The run-time collision detection algorithm of SWIFT++ follows the basic culling strategy of BVHs. The query between two convex hulls is performed using an extension of the Voronoi marching algorithm [LC91]. This allows finding the closest features between convex patches, and computing distance information. SWIFT++ also exploits properties of convex hulls to enable early exit in intersection tests. When some convex hulls in the BVHs intersect, there is no need to continue with the recursive BV tests if the intersecting triangles lie on the original surfaces. Finally, SWIFT++ exploits motion coherence by caching the closest primitives at every pair of tested convex hulls in the previous frame, and starts Voronoi marching from those primitives, achieving $\mathrm{O}(1)$ query time for each pair of convex hulls in most situations.

17.1.3. 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 randomised 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 [Ber01] furthered his work. Kim et al. [KLM02a] presented DEEP, 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 discretisation. 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. [KLM02b] 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.

DEEP: Dual-Space Expansion for Estimating Penetration Depth

The DEEP algorithm by Kim et al. [KLM02a] aims at computing the penetration depth between two convex polytopes. It exploits the definition of penetration depth as the minimum distance from the origin to the boundary of the Minkowski sum (or configuration space obstacle, CSO) of two objects A and $-B$.

In practice, DEEP computes implicitly the surface of the Minkowski sum by constructing local Gauss maps of the objects. For a pair of convex polytopes to be tested, the Gauss maps are precomputed. At runtime, and using a pair of initialisation features, the Gauss map of one object is transformed to the local reference system of the other object, the Gauss maps are projected onto a plane, and the arrangement is implicitly computed. Vertices on the Gauss map correspond to triangles in the polytopes, edges correspond to edges, and faces correspond to vertices. The features that realise the penetration depth also define a penetration direction, which corresponds to a certain direction in the intersected Gauss maps. The problem of finding the penetration features reduces to checking distances between vertex-face or edge-edge features that overlap in the Gauss map, as these are the features that define the boundary of the Minkowski sum.

DEEP proceeds by walking to neighboring feature pairs that minimise the penetration depth, until a local minimum is reached. The distance from the origin to the boundary of the Minkowski sum of two convex polytopes may present several local minima, but DEEP reaches in practice the extreme minimum if appropriate initialisation features are given. In situations with high motion coherence, the penetration features from the previous frame are usually good initialisation features.

Kim et al. [KLM02a,KLM04] performed evaluation tests on DEEP, and they found that the query time in situations with high motion coherence is almost constant, and the performance is better than previous algorithms [Ber01], both in terms of query time and reaching the extreme minimum.

DEEP has been integrated with SWIFT++ [EL01] for handling the penetration depth between non-convex objects. First, SWIFT++ handles the hierarchical test using BVHs of convex hulls, and in case leaf convex hulls intersect, DEEP computes the penetration depth and the features that realise the penetration depth. The combination of SWIFT++ and DEEP has also been tested for performing collision detection for haptic rendering [KOLM03].

17.1.4. Multi-Resolution Collision Detection

Multi-resolution 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 multi-resolution analysis [SDS96].

Multi-resolution 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 multi-resolution 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). Multi-resolution 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 multi-resolution 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.

Multi-resolution collision detection refers to the execution of approximate collision detection queries using adaptive object representations. Hubbard [Hub94] introduced the idea of using sphere-trees [Qui94] for multi-resolution 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 minimise 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 multi-resolution representations for haptic rendering has also been investigated by several researchers. Pai and Reissel [PR97] investigated the use of multi-resolution image curves for 2D haptic interaction. El-Sana and Varshney [ESV00] applied LOD techniques to 3-DoF haptic rendering. They created a multi-resolution 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. Otaduy and Lin [OL03b,OL03a] introduced *contact levels of detail*, dual hierarchical representations for multi-resolution collision detection, and they applied them to 6-DoF haptic rendering, producing a sensation-preserving simplified rendering. Otaduy et al. [OJSL04] also introduced a representation based on *haptic textures* and low-resolution geometric models, along with a perceptually-driven force model, for haptically rendering the interaction between textured objects. Contact levels of detail and haptic textures are described in depth in §18 and §19 respectively.

17.1.5. Other Techniques for Collision Detection

We 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. Rasterisation 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 inter-frame 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.

17.2. 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**:

$$\begin{aligned} \mathbf{F}(t) &= \dot{\mathbf{P}}(t) = m\,\ddot{\mathbf{x}}(t), \\ \mathbf{T}(t) &= \dot{\mathbf{L}}(t) = \omega(t) \times (M\omega(t)) + M\dot{\omega}(t). \end{aligned} \tag{3}$$

As shown in the equations, momentum derivatives can be expressed in terms of the linear acceleration of the centre 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 we 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.

17.2.1. 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 modelled 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:

$$\begin{aligned} \mathbf{F}_A &= -f(\delta)\mathbf{n}, \\ \mathbf{T}_A &= (\mathbf{p} - \mathbf{c}_A) \times \mathbf{F}_A, \end{aligned} \tag{4}$$

where $\mathbf{c}_A$ is the centre 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 §17.1, 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, OL05] 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, OL05].

Friction effects can be incorporated into penalty-based methods by means of localised 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 modelled 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, we have analysed 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 un-stretched. In other words, the penetration depth value must be zero initially and must grow smoothly. 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.

17.2.2. 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. 3), subject to geometric constraints that prevent object interpenetration. The numerical integration of Newton-Euler equations must be interrupted before objects inter-penetrate. 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:

$$\begin{aligned} \mathbf{a} &= A\mathbf{F} + \mathbf{b}, \\ \mathbf{a} &\geq 0, \quad \mathbf{F} \geq 0. \end{aligned} \tag{5}$$

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:

$$\min\left(\mathbf{F}^T\mathbf{a}\right) = \min\left(\mathbf{F}^T A\mathbf{F} + \mathbf{F}^T\mathbf{b}\right). \tag{6}$$

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 [Lot84]. 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 practise.

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 pre-stretched 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 [Mir98b] describes polygonal contact areas as combinations of edge-edge and vertex-face contacts.

17.2.3. 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 parameterised separately, and numerical integration is performed in order to compute the velocities after the collision. The parameterisation 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 micro-collisions 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.

17.3. Techniques for Six-Degree-of-Freedom Haptic Rendering

The problem of 6-DoF haptic rendering has been studied by several researchers. As introduced in §16.2.1, the existing methods for 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. We 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. To conclude the section we focus on the rendering pipeline suggested by Otaduy and Lin [OL05], combining virtual coupling, penalty-based collision response, implicit integration of rigid-body dynamics, and the perceptually-driven collision detection methods described in further sections.

17.3.1. 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, localised computation of per-contact penetration depth [KLM02a]. 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 realise 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.

17.3.2. 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 destabilise 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.

17.3.3. 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 §17.2, 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 §17.2.1, 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 inter-penetrations 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 demonstrated by Otaduy and Lin [OL05], implicit integration can be performed at force update rates under the assumption that only the grasped object is dynamic.

17.3.4. Virtual Coupling with Multi-Resolution Techniques

The application of 6-DoF haptic rendering algorithms to complex models and complex contact scenarios becomes a challenging issue, due to the inherent cost of collision detection that induces slow force updates. Otaduy and Lin [OL05] have integrated perceptually-driven multiresolution collision detection techniques [OL03b, OJSL04] with a virtual-coupling-based pipeline, thus achieving stable and responsive 6-DoF haptic rendering of complex models.

In order to achieve a wide range of impedances, and thereby stable and responsive rendering, Otaduy and Lin simulate the rigid body dynamics of the grasped object using implicit integration and penalty-based collision response. On the one hand, implicit integration of rigid body dynamics provides passivity and higher stability under a larger combination of mass and stiffness values. Low mass values enable low coupling impedance during free-space motion, whereas high stiffness values enable high coupling impedance in contact situations. On the other hand, penalty-based collision response is fast to compute, enabling fast simulation of rigid body dynamics and a high force update rate, with beneficial results on the stability and responsiveness of the interaction as well. Penalty-based methods cannot enforce non-penetration. But, by allowing small collision tolerances and using high contact stiffness values, the resulting object interpenetration becomes almost imperceptible when using penalty-based methods.

By using a linearised contact model, Otaduy and Lin decompose the haptic rendering pipeline into two threads: a haptic thread that performs the rigid-body dynamic simulation of the grasped object, and a contact thread that executes collision detection and response. The linearised contact model decouples the process of collision detection from the simulation of rigid body dynamics, while still enabling a very fast update of collision forces in the simulation. In this way, collision detection is less a bottleneck for the simulation and the synthesis of feedback force and torque. Moreover, the use of perceptually-driven multiresolution collision detection techniques maximises the update rate of collision detection.

The rendering pipeline proposed by Otaduy and Lin [OL05] bears some similarities with other existing approaches in 6-DoF haptic rendering, in the sense that it borrows the concepts of virtual coupling and intermediate representations widely used by others. The generality of the pipeline would also allow for interchangeable collision detection techniques, employing for example the voxel-based approach suggested by McNeely et al. [MPT99] or the spatialized normal cone hierarchies by Johnson and Cohen [JC01]. Moreover, these different collision detection techniques could potentially exploit the perceptual observations that have driven the multiresolution collision detection techniques described in the following sections.

18. Sensation Preserving Simplification of Complex Geometry

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, existing exact 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. 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.

This section describes *contact levels of detail* (CLODs), a multi-resolution collision detection algorithm that integrates BVHs and LODs in one single *dual hierarchy*. We 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 multi-resolution collision detection and select the appropriate object resolution at each contact location, as shown in Figure 46. Findings from tactual perception and spatial recognition [KL95, OC99, OC01] demonstrate that large contact areas reduce the perceptibility of fine surface features.

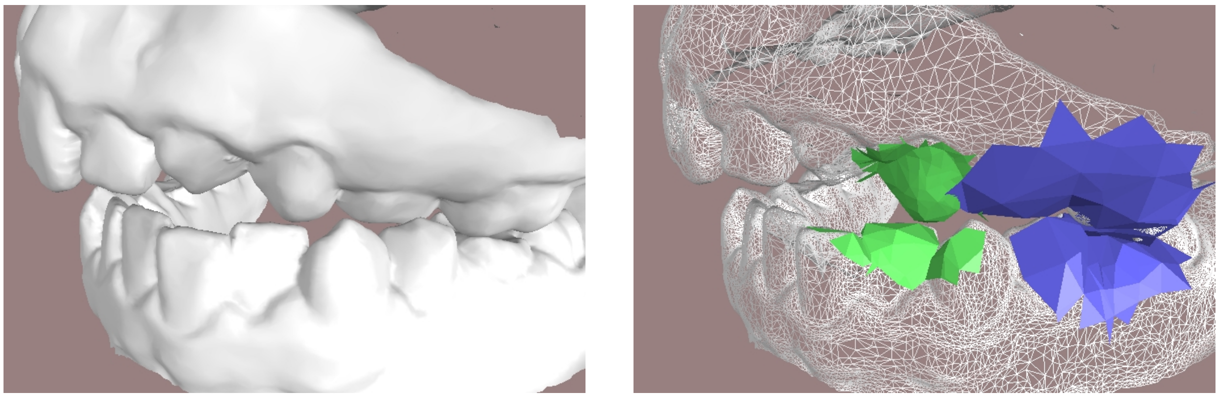


Figure 46: ***Multi-resolution Collision Detection Using CLODs.*** Left: Moving jaws in contact, rendered at their highest resolution. Right: The appropriate object resolution (shown in blue and green) is adaptively selected at each contact location, while the finest resolution is displayed in wireframe.

The practical implementation of CLODs involves many design decisions. One of them is the type of bounding volume (BV) for the BVH. Otaduy and Lin [OL03b] suggested convex hulls as the BVs for implementing 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, the atomic operation *filtered edge collapse* performs mesh decimation and filtering subject to convexity constraints.

At runtime, multi-resolution 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. Otaduy and Lin [OL03b] presented 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.

CLODs have been applied to both 6-DoF haptic rendering and rigid body simulation. In 6-DoF haptic rendering of challenging contact scenarios using CLODs, Otaduy and Lin [OL03b] observed up to 2-orders-of-magnitude performance improvement over exact collision detection methods with little degradation in the contact forces.

This section compiles work and results previously published in [OL03b] and [OL03a], and it is organised as follows. §18.1 presents the motivation and design goals of a multi-resolution collision detection algorithm for haptic rendering. §18.2 introduces the data structure of CLODs, and §18.3 presents a particular implementation based on convex hulls, followed by the description of sensation preserving simplification. §18.4 explains how contact levels of detail are used in runtime collision detection. §18.5 presents the experiments and results and, last, §18.6 concludes with a discussion of limitations.

18.1. Foundations and Objectives of Contact Levels of Detail

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

18.1.1. Haptic Perception of Surface Detail

§16.3.1 summarised 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 §18.3) 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 neighbourhood 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.

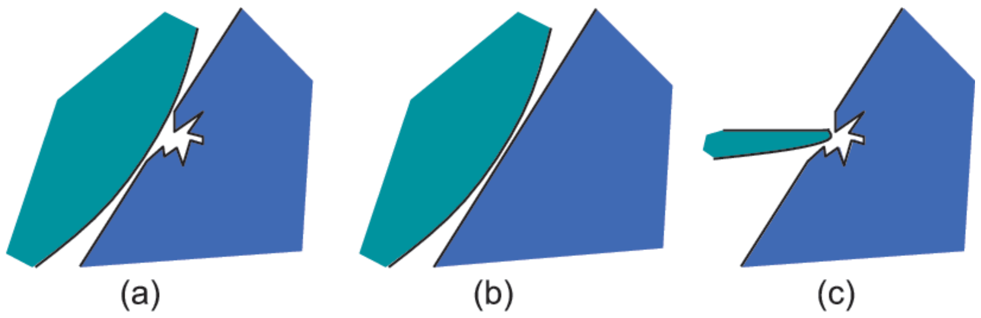


Figure 47: ***Contact area and resolution:*** (a) high-resolution model with large contact area; (b) low-resolution model with large contact area; (c) high-resolution model with small contact area.

Figure 47 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. Haptic error metrics 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. In §18.3, we show how haptic error metrics drive the offline construction of CLODs, and in §18.4, we show how they are used in runtime contact queries.

18.1.2. 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, characterised 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 §17.1 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 §17.1.2, 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 initialising 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 modelled; higher resolutions imply a deeper BVTT. To summarise, 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 §18.1.1, 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 multi-resolution representations of the objects and selecting the appropriate level of detail (i.e., resolution) for each object at each contact location independently.

18.1.3. Design Requirements and Desiderata

Efficient multi-resolution collision detection depends on two main objectives:

1. Create **accurate multi-resolution representations**.
2. Embed the multi-resolution representations in **effective bounding volume hierarchies**.

Multi-resolution 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, *contact levels of detail* constitute a unique dual hierarchical representation, which serves as both a multi-resolution representation and a BVH.

On the one hand, this dual hierarchy constitutes a multi-resolution 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.

§18.2 describes the generic data structure employed in CLODs. The implementation of CLODs, however, requires the selection of one particular type of bounding volume. §18.3 presents the construction of CLODs using convex hulls as the bounding volumes.

To summarise, CLODs address the goal of creating **dual multi-resolution hierarchies** that:

1. **Minimise perceptible surface deviation.** This goal is achieved 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.** A surface convex decomposition is performed on the given triangular mesh, and it is maintained 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).

18.2. Data Structure

Prior to describing the data structure for CLODs, we introduce some notation to be used throughout this section. Then we describe the data structure, discuss its interpretation as a multi-resolution representation, and overview the process of building generic CLODs.

18.2.1. Notation

In the remaining of this section, we use bold-face letters to distinguish a vector (e.g., a point, normal, etc.) from a scalar value. In Table 3, we enumerate some of the notations used throughout the section.

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: ***Notation Table***

18.2.2. 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 $\{c_{i,0}, c_{i,1}, ..., c_{i,m}\}$.

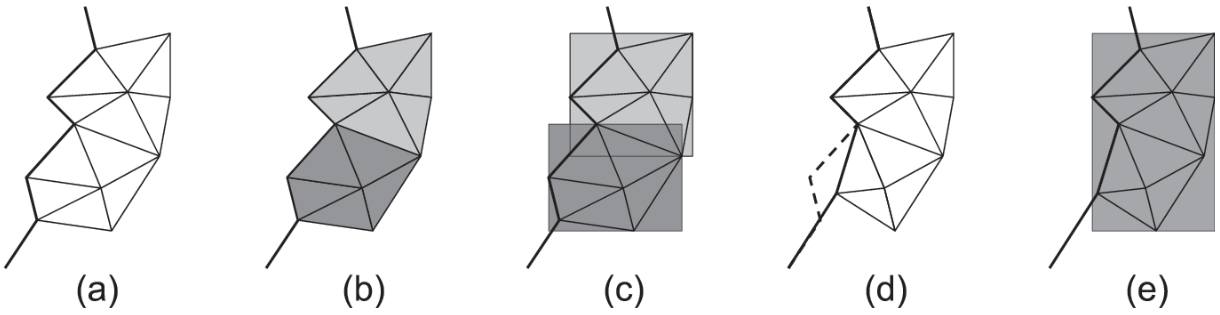


Figure 48: ***Construction of generic CLODs:*** (a) Initial surface; (b) Clusters of triangles; (c) BVs for each cluster; (d) Mesh simplification; (e) BV of the union of clusters after some conditions are met.

- 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 §18.4.2.

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.

Multi-resolution Interpretation

The CLODs of a given object comprise a multi-resolution 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.

18.2.3. Generic Construction of Contact Levels of Detail

The process of creating the CLODs, depicted in Figure 48, 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 optimised. The next step in the creation of CLODs is to compute the BV of each cluster. This initialisation 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 §18.3, we present the sensation preserving simplification process that results in a hierarchy of CLODs of convex hulls. We also describe the atomic simplification operations and the constraints imposed by the selection of convex hulls as the BVs.

18.3. Sensation Preserving Simplification Process

In this section we 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. We first discuss the selection of convex hulls as the bounding volumes and we define the concept of resolution in the context of CLODs. Next, we present the detailed process of sensation preserving simplification, followed by a description of the atomic simplification operation *filtered edge collapse*. We end this section presenting some examples of CLODs.

18.3.1. Selection of Convex Hulls as Bounding Volumes

Overlap tests between convex hulls can be executed in expected constant time with motion coherence [LC91, Mir98a, 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].

The clusters of the initial mesh M_0 are defined as the surface patches of its convex surface decomposition [CDST97, EL01], in order to maximise the efficiency of runtime collision detection using convex hulls as BVs. Otaduy and Lin [OL03b] 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 decomposition requires the input models to be described as 2-manifold, oriented triangle meshes.

18.3.2. Definition and Computation of Resolution

Before explaining how to generate each LOD, we define *resolution* in the context of CLODs. The definition follows the framework of signal processing for irregular meshes and assumes 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.

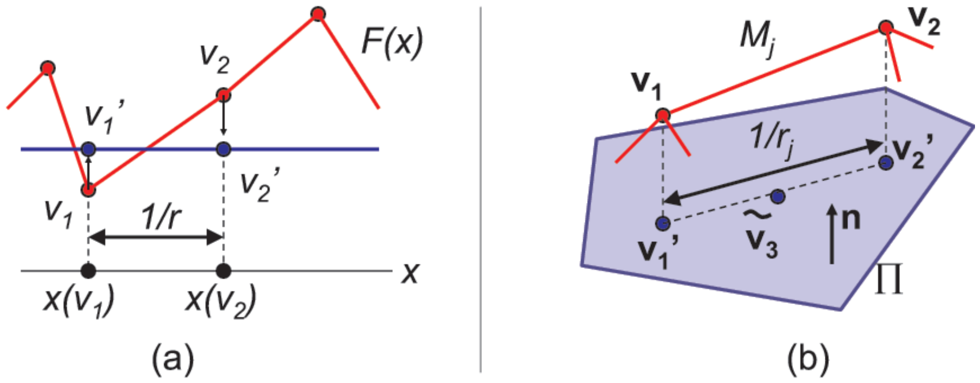


Figure 49: ***Definition of Resolution.*** (a) Resolution r defined in the 1D setting; (b) Resolution for irregular meshes.

The 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 Figure 49-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 Figure 49-b.

The low-resolution mesh M_{j+1} is computed locally by filtering the mesh M_j, applying the filtered edge collapse operation to the edge $(\mathbf{v}_1, \mathbf{v}_2)$. Then, the normal $\mathbf{n}$ of the resulting vertex $\tilde{\mathbf{v}}_3$ is computed 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.

$$r = \frac{1}{\|(\mathbf{v}_1 - \mathbf{v}_2) - ((\mathbf{v}_1 - \mathbf{v}_2)\cdot\mathbf{n})\cdot\mathbf{n}\|}. \tag{7}$$

18.3.3. Construction of Contact Levels of Detail

The construction of CLODs of convex hulls is initialised 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 neighbouring 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. A new LOD is output 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 practise, 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 multi-resolution collision queries, but are necessary to complete the BVH.

Algorithm 18.1 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:

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

Edge collapse is employed 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 down-sampling operation, in which two samples (i.e., vertices) are merged into a single one.

There are many possible approaches for prioritising edges and selecting the position of the resulting vertices: minimisation of energy functions [Hop96], optimisation approaches [LT98], quadric error metrics for measuring surface deviation [GH97a], and more. None of these approaches, however, meets the convexity constraints or takes into account the factors that maximise 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 multi-resolution representations of the initial mesh, because they undergo an enlargement process that can result in noticeable visual artifacts. As an alternative, Otaduy and Lin [OL03b] suggest the local simplification operation *filtered edge collapse*, inspired by multi-resolution analysis and signal processing of meshes.

Resolution and the Simplification Process

As discussed in §18.2.2, 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, edges to be collapsed are prioritised based on their resolution.

In the construction of CLODs, and as part of the initialisation, the resolution of all edges is computed, they are set as valid for collapse, and inserted in a priority queue. At each simplification step, the edge with highest priority (i.e., highest resolution)

```
Compute surface convex decomposition
$n$ = number of convex patches
Compute resolution of edges
Output initial LOD
Initialise 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 $\leq n/2$ then
        Output new LOD
        $n$ = number of convex patches
    endif
endwhile
while Number of patches $> 1$,
    Binary merge of patches
endwhile
```

ALGORITHM 18.1: Pseudo Code of the Sensation Preserving Simplification Loop

is selected for collapse. If the 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 maximise the resolution at which LODs are generated. As explained in §18.4, the perceptual error for haptic rendering is measured by taking into account the resolution of the surface detail that is culled away. Multi-resolution contact queries will terminate faster as a result of maximising the resolution at which LODs are generated. This is the basis for selecting edge resolution as the priority for edge collapses.

18.3.4. Filtered Edge Collapse

The construction of CLODs aims to:

1. Generate multi-resolution 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 down-sampling 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 $\hat{\mathbf{v}}_3$. This step provides the down-sampling.
2. An initialisation process that sets the position of $\hat{\mathbf{v}}_3$ using quadric error metrics [GH97a].
3. Unconstrained relaxation to a position $\tilde{\mathbf{v}}_3$, using Guskov's minimisation of second order divided differences [GSS99].
4. The solution of an optimisation problem in order to minimise 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 minimisation of dihedral angles, without much affecting the shape of the triangles [GSS99]. Other filtering techniques, such as those proposed by Taubin [Tau95], provide similar results, but 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 minimisation of second order differences. Next, we 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 neighbourhood 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 $\mathbf{v}_3$.

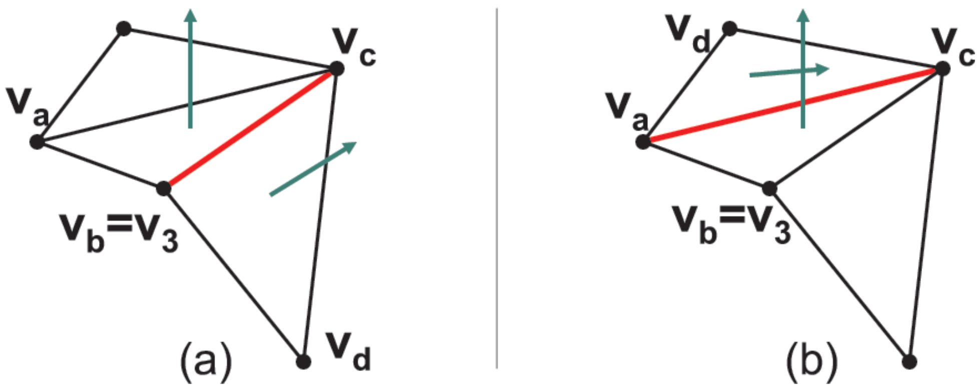


Figure 50: ***Local Convexity Constraints.*** Assignment of vertices $\mathbf{v}_a$, $\mathbf{v}_b$, $\mathbf{v}_c$ and $\mathbf{v}_d$ for an interior edge (a) incident on $\mathbf{v}_3$, and (b) opposite to $\mathbf{v}_3$.

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 Figure 50. However, both cases can be treated equally. Assigning $\mathbf{v}_a$, $\mathbf{v}_b$, $\mathbf{v}_c$ and $\mathbf{v}_d$ vertices as in Figure 50, the convexity constraint of an edge can be expressed as a negative volume for the parallelepiped defined by the adjacent triangles:

$$((\mathbf{v}_b - \mathbf{v}_a) \times (\mathbf{v}_c - \mathbf{v}_a)) \cdot (\mathbf{v}_d - \mathbf{v}_a) \leq 0. \tag{8}$$

The convexity constraints can be satisfied by formulating an optimisation program in which $\mathbf{v}_3$ is constrained to the segment between the position that minimises surface deviation, $\hat{\mathbf{v}}_3$, and the unconstrained filtered position, $\tilde{\mathbf{v}}_3$. The objective function is the distance to $\tilde{\mathbf{v}}_3$. 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:

$$\begin{aligned} \mathbf{v}_3 &= u \cdot \hat{\mathbf{v}}_3 + (1-u) \cdot \tilde{\mathbf{v}}_3, \\ u &\geq 0, \\ u &\leq 1. \end{aligned} \tag{9}$$

The convexity constraints in Eq. 8 can be rewritten based on constants A and B as:

$$\begin{aligned} A \cdot u + B \geq 0, &\quad \text{where} \\ A = ((\mathbf{v}_d - \mathbf{v}_a) \times (\mathbf{v}_c - \mathbf{v}_a)) &\cdot (\hat{\mathbf{v}}_3 - \tilde{\mathbf{v}}_3), \\ B = ((\mathbf{v}_d - \mathbf{v}_a) \times (\mathbf{v}_c - \mathbf{v}_a)) &\cdot (\tilde{\mathbf{v}}_3 - \mathbf{v}_a). \end{aligned} \tag{10}$$

The resulting vertex $\mathbf{v}_3$ corresponds to the minimum value of u that meets all the constraints. When $\tilde{\mathbf{v}}_3$ 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 optimisation program, so they cannot be incorporated into the filtering process. Instead, they have to be verified after the filtering has been performed. They are verified 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, the global constraints are checked. If they are met, the edge collapse is valid. If they are not met, then the global constraints are checked at $\hat{\mathbf{v}}_3$. If they are not met at $\hat{\mathbf{v}}_3$ either, the edge collapse is considered invalid and it is disabled. If $\hat{\mathbf{v}}_3$ meets the global constraints, a bisection search is performed between $\hat{\mathbf{v}}_3$ and $\mathbf{v}_3$ of up to K iterations (in the practical implementation $K = 3$), searching for the position closest to $\tilde{\mathbf{v}}_3$ that meets the global convexity constraints, as shown in Figure 51. The resulting vertex $\mathbf{v}_3$ is reassigned to this position.

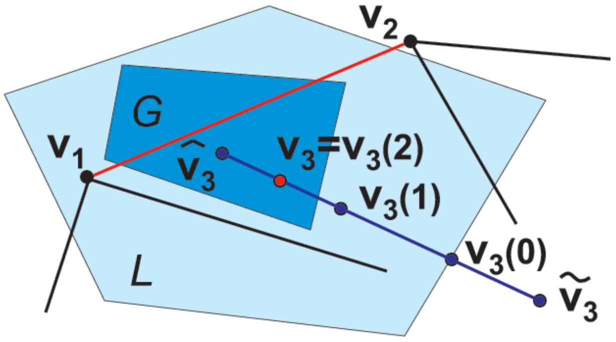


Figure 51: ***Filtered Edge Collapse with Convexity Constraints.*** The figure shows a filtered edge collapse in which bisection search is required to find a position that meets the convexity constraints. G and L represent feasible regions of global and local constraints respectively.

18.3.5. Parameters for Error Metrics

To perform multi-resolution collision detection for haptic rendering, one must define error metrics that will dictate the selection of CLODs. The error metrics are described in §18.4, but here we enumerate the parameters that have to be computed after performing sensation preserving simplification and constructing the CLODs.

Besides the resolution r of each LOD, the error metrics require 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, one may use the same distance tolerance as the one used for contact queries (see §18.4). When an edge $(\mathbf{v}_1, \mathbf{v}_2)$ is collapsed to a vertex $\mathbf{v}_3$, the support area of $\mathbf{v}_3$ is set as 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.

18.3.6. Examples of Contact Levels of Detail

Figure 52 shows several of the LODs obtained when processing a model of a lower jaw (see §18.5 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 multi-resolution 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

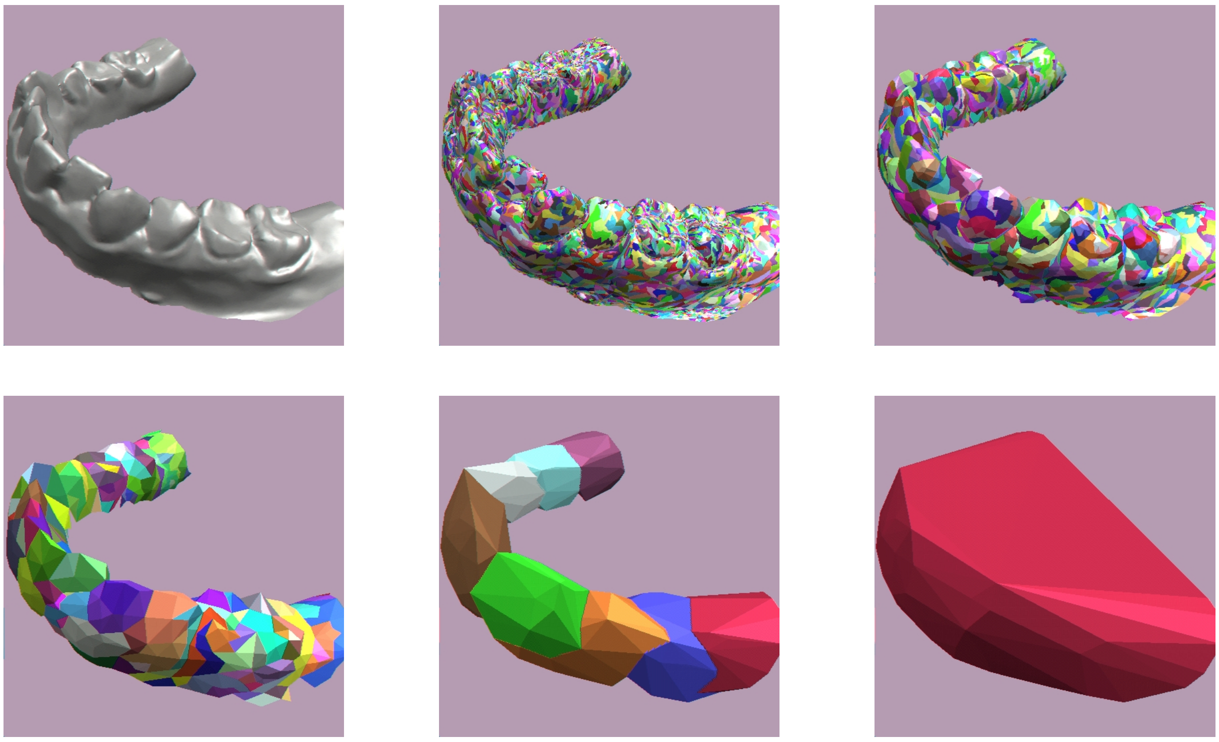


Figure 52: ***CLODs of a Lower Jaw.*** From left to right and top to bottom, original mesh, M_0, and convex patches of M_0, M_3, M_6, M_{11}, and M_{14}.

necessarily bound the original surface, as may be deduced from the figures. As described later in §18.4.2, 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 multi-resolution hierarchy.

Figure 53 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.

18.4. Multi-Resolution Collision Detection

In the previous section we have described the creation of CLODs. Ultimately, CLODs are intended to be used at runtime collision detection. Although here we focus on 6-DoF haptic rendering, CLODs can also be used to accelerate contact queries in rigid body simulation.

In this section we describe a multi resolution collision detection algorithm based on CLODs, as well as associated selective refinement criteria. We first introduce the type of contact queries relevant to 6-DoF haptic rendering, and then we describe multi resolution collision detection using CLODs. We also present error metrics and define the selective refinement tests for both haptic rendering and rigid body simulation. To conclude this section, we discuss how contact information from CLODs can be used for collision response in haptic rendering and rigid body simulation.

18.4.1. 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.

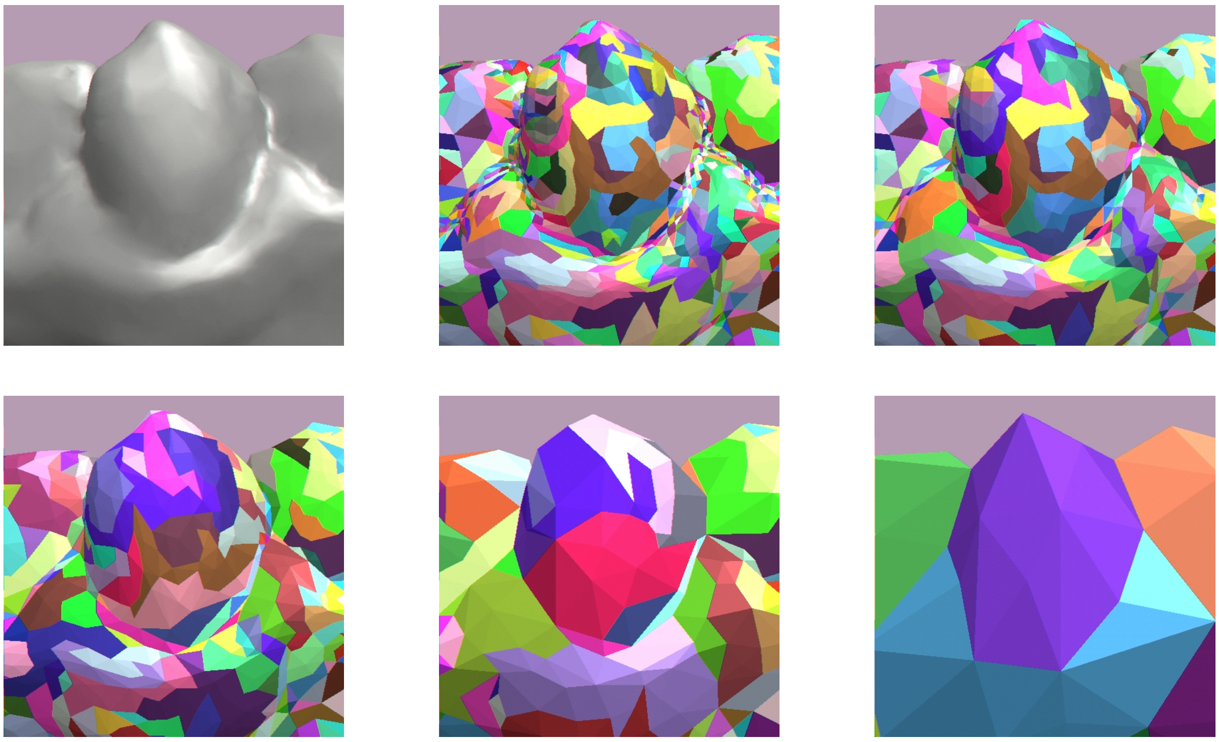


Figure 53: ***Detail View of the CLODs of a Lower Jaw.*** From left to right and top to bottom, original mesh, M_0, and convex patches of M_0, M_1, M_2, M_4, and M_7.

For haptic rendering purposes, we 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, $Q(A,B,d)$ is solved by performing a surface convex decomposition of A and B [EL01] and testing pairwise distances between convex BVs [GJK88, Lin93, Cam97, Mir98a, EL00]. The distance query $q(a,b,d)$ between two convex pieces $a \in A$ and $b \in B$ is defined 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 localised penetration depth values [KOLM03].

As mentioned in §18.1.2, 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.

18.4.2. Multi-Resolution Contact Query

Using CLODs, multi-resolution collision detection can be implemented by slightly modifying the typical collision detection procedures based on BVHs. In multi-resolution 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 Figure 46. 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 §18.4.3, we describe the test for selective refinement in more detail, and present 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:

$$d_{ab} = d + h(a) + h(b), \tag{11}$$

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 §18.3.5, 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 analyse 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, the BV whose children have coarser resolution is split. This splitting policy yields a BVTT in which the levels of the tree are sorted according to their resolution, as shown in Figure 54. Nodes of the BVTT at coarser resolution are closer to the root. A resolution-based ordering of the BVTT is a key factor for maximising 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 §18.3.2 is an object-independent absolute metric. If objects are scaled, the value of resolution must be scaled accordingly.

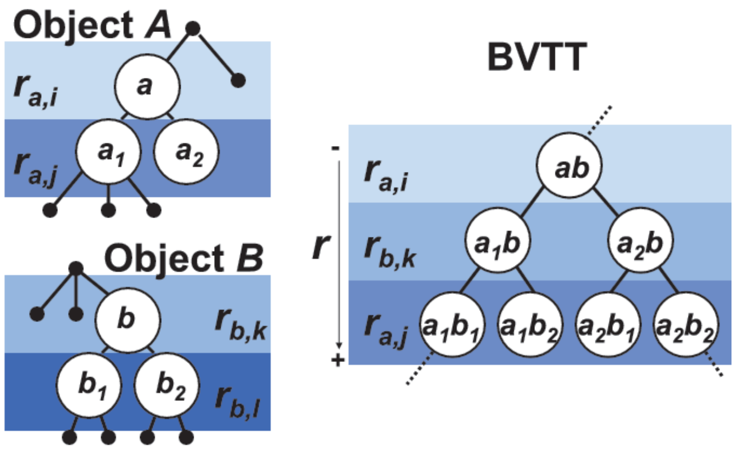


Figure 54: ***Resolution-Based Ordering of the Bounding Volume Test Tree.*** A node splitting policy based on CLOD resolution implies a BVTT in which levels are sorted according to increasing resolution. Descending on the BVTT has the effect of increasing CLOD resolution.

As pointed out in §18.3.3, 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 Figure 55 by the line λ.

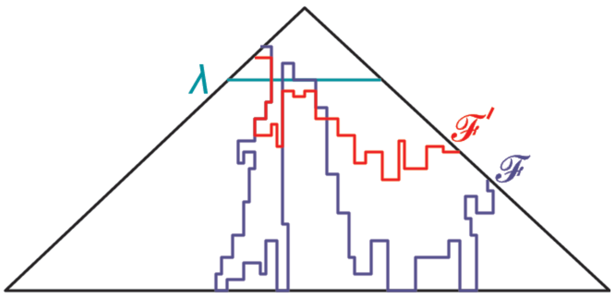


Figure 55: ***Generalised Front Tracking of the BVTT.*** The front of the BVTT for an exact contact query, $\mathbb{F}$, is raised up to the new front $\mathbb{F}'$ using CLODs, since the recursive distance queries can stop at coarser resolutions. λ indicates the free CLODs, constructed by unconstrained merging of convex patches.

As indicated in §18.1.2, temporal coherence can be exploited using GFT. One can store the front $\mathbb{F}$ of the BVTT where the result of the distance query q switches from true to false, as shown in Figure 55. 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 $\mathbb{F}$. 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 $\mathbb{F}$. 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 multi-resolution collision detection setting, the addition of a selective refinement test further increases the performance of the query. In the BVTT, the new active front, $\mathbb{F}'$, is above the original front $\mathbb{F}$ 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 Figure 55. This approach results in a much faster processing of contact queries and, ultimately, it enables 6-DoF haptic rendering of complex objects.

18.4.3. Selective Refinement and Error Metrics

As discussed in §18.1.1, the perceptibility of surface features depends on the ratio between their size and the contact area. Following this observation, Otaduy and Lin [OL03b] 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:

$$\phi_a = \frac{s_a}{r_a^2}, \tag{12}$$

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, a weighted surface deviation s^* is computed as:

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

$$D = \max(D_a, D_b). \tag{13}$$

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 §18.3.5, a support area D is stored for every vertex in the CLOD data structure. For edge or triangle contact primitives, one may 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 we 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. We 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, s_0 can be defined 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 §18.5. Based on experiments described in §18.5.2, 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,

$$s_0 = \frac{N \cdot l \cdot f}{n \cdot i}. \tag{14}$$

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 prioritise 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 prioritised based on $\frac{s^*}{s_0}$.

18.4.4. Solving Discontinuities in Collision Response

A major issue in systems that use multi-resolution representations is the discontinuity that arises when the algorithm switches between different LODs. This problem is known as "popping" in multi-resolution (visual) rendering.

In multi-resolution 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, we discuss some interpolation techniques to resolve discontinuities in each of these cases.

Interpolation of Contact Information

A common approach for 6-DoF haptic rendering is to compute 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, a conservative refinement step is performed, and contact information is computed for the children of the current node of the BVTT. The contact information is then 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 §18.5.

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 non-existent collision event in the interval $[t_i, t_{i+1}]$.

This problem can be solved 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.

18.5. Experiments and Results

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

18.5.1. Benchmark Models

Table 4 shows statistics of CLOD representations for a list of models. 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 normalised resolution (for unit object radius) of the finest and coarsest CLODs, and the number of "free" and total CLODs. As described in §18.3.3, 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

Table 4: ***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 $1,468$ 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 Figures 57, 58, and 59.

18.5.2. 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.

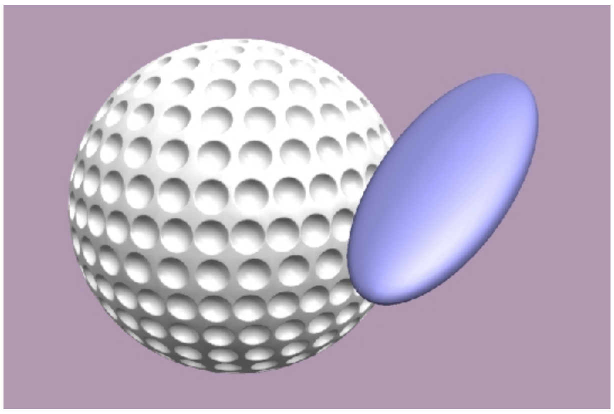


Figure 56: ***Exploration of a Multi-resolution Golf Ball with an Ellipsoid.*** Scenario of the experiments for identifying haptic error tolerances with CLODs.

Otaduy and Lin [OL03b] conducted an 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. 4 for statistics of the model) that is explored with an ellipsoid, as shown in Figure 56. 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, a CLOD representation is created only for the golf ball, and the ellipsoid is left 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.

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

Table 5: ***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.

Table 5 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 the models. The users also reported that the main characteristic they explored was the perceptibility of the dimples of the golf ball.

18.5.3. Performance Experiments in 6-DoF Haptic Rendering

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

- Moving upper and lower jaws (See Figure 57).
- Interlocking ball joints (See Figure 58).

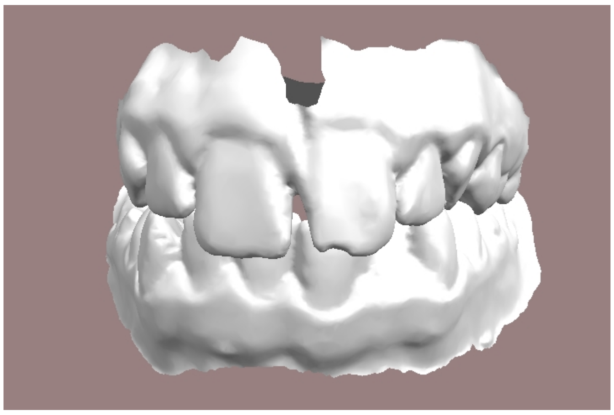


Figure 57: ***Upper and Lower Jaws.*** A benchmark scenario for 6-DoF haptic rendering using CLODs.

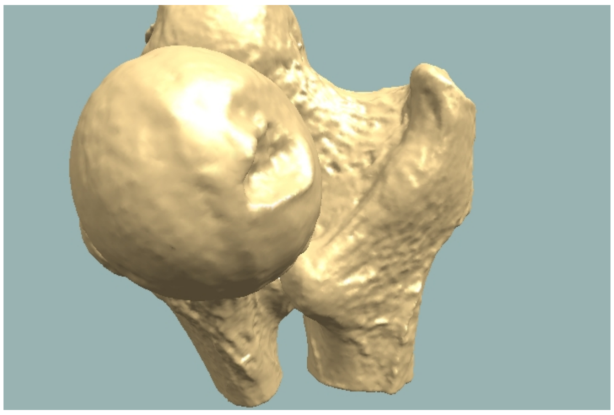


Figure 58: ***Ball Joints.*** A benchmark scenario for 6-DoF haptic rendering using CLODs.

- Golf club tapping a golf ball (See Figure 59).

Statistics of the CLOD data structures of the models have been given in Table. 4.

Contact forces and running time are analysed on the benchmarks of the moving jaws and the golf club and ball. In particular, force profiles and statistics of the contact query are compared 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 [KLM02a]. In the graphs shown later, these simulations are referred to 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, we describe implementation details and the performance results.

Implementation Details

The haptic demonstrations were performed using a 6-DoF PHANTOM 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, was developed using C++. The implementation of multi-resolution collision detection based on CLODs uses distance and penetration depth queries between convex patches from the publicly available libraries SWIFT++ [EL01] and DEEP [KLM02a].

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. The experiments employed the direct haptic rendering pipeline described

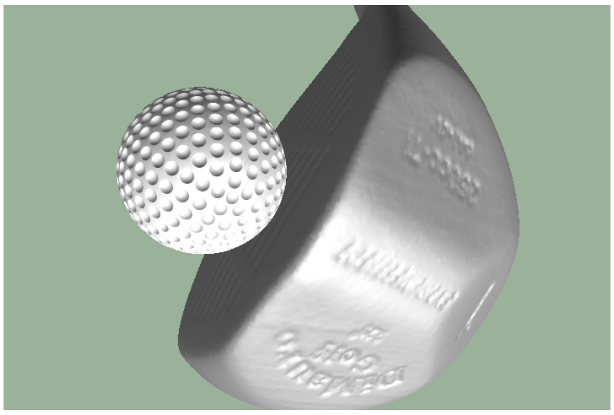


Figure 59: ***Golf Club and Ball.*** A benchmark scenario for 6-DoF haptic rendering using CLODs.

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 stabilising 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. For higher stability, the output of collision response may be integrated in a more stable haptic rendering pipeline such as the one presented in [OL05].

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

Performance Results and Analysis

Figure 60 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 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. 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.

Figure 61 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 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 Figure 60 and Figure 61 are due to context switching in the CPU.

From the analysis of the contact profiles, one can 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.

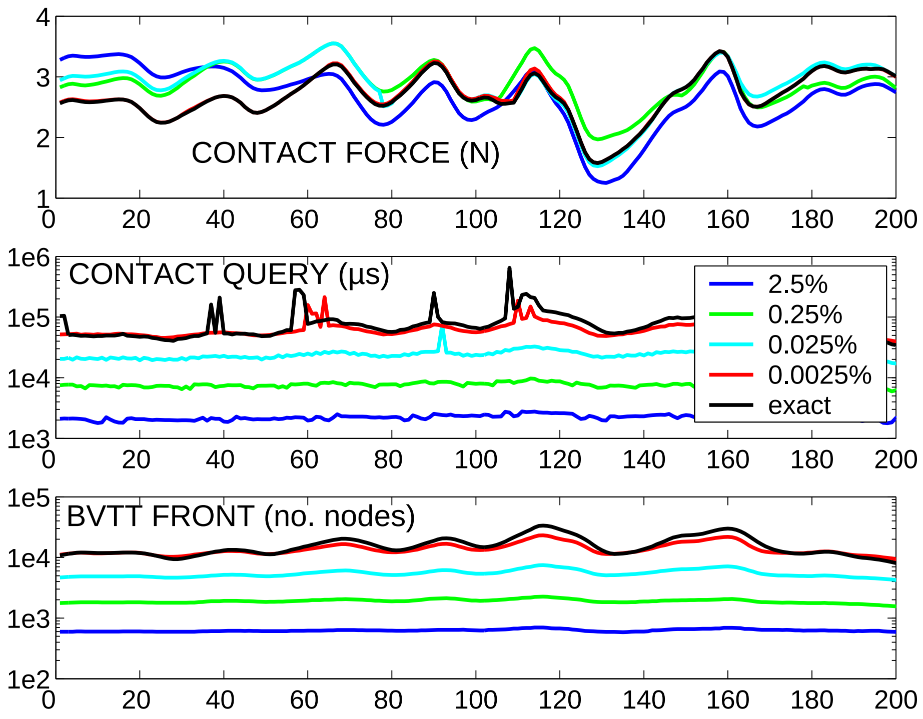


Figure 60: ***Contact Profile for Moving Jaws.*** Top: The profiles of the contact forces displayed using CLODs, with varying error tolerances up to 2.5% of the radii of the jaws, all show very similar patterns. This similarity implies that the sensations of shape provided to the user are nearly identical.
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.

- 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 multi-resolution approach enables force update rates suitable for interactive haptic rendering.

18.6. Discussion and Limitations

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

18.6.1. 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.

18.6.2. Limitations Related to the Construction of CLODs

From the perspective of constructing a multi-resolution representation, sensation preserving simplification can be compared with both mesh decimation techniques and mesh filtering techniques.

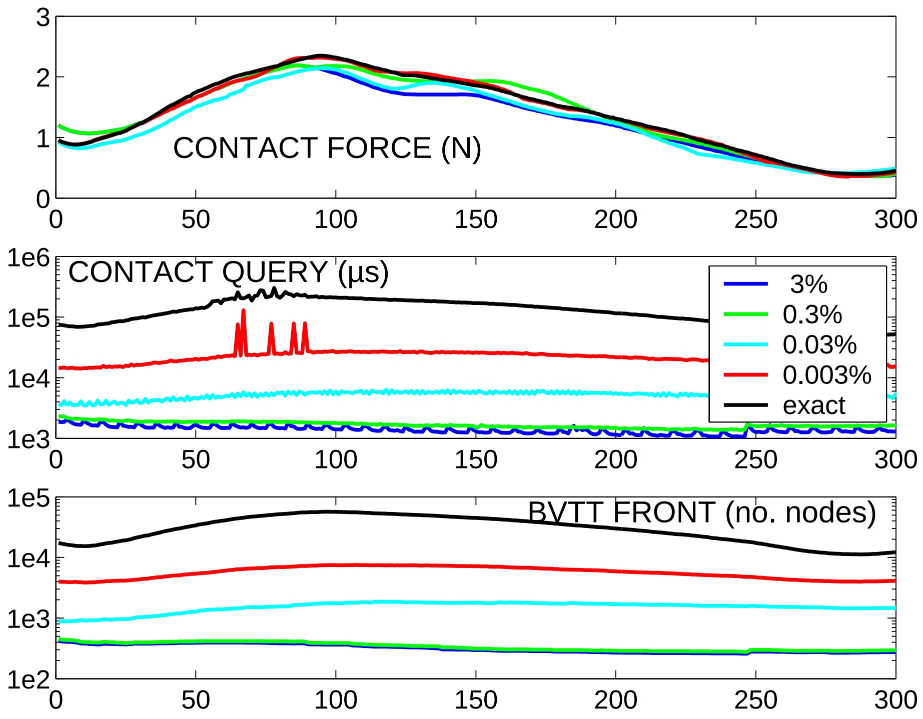


Figure 61: ***Contact Profile for Golf Scene.*** Top: The profiles of the contact forces displayed using CLODs, with varying error tolerances up to 3% of the radius of the ball, show nearly identical patterns.
Middle: A *log* plot of contact query time using CLODs with various error tolerances shows 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.

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, GH97a, 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 minimise 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 multi-resolution collision detection, and these are worth exploring too.

- **Lack of containment.** As introduced in §18.2.2, in CLODs, a level of the multi-resolution representation may not bound the original surface. This has two drawbacks: (1) it requires a modification of the contact queries, as explained in §18.4.2, and (2) it implies that multi-resolution 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 §18.3.3, 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 §18.3.3, the BVH may contain some levels that cannot be used to report multi-resolution contact information, because they cannot be considered as low-resolution representations of the input object. The existence of free CLODs reduces the applicability of multi-resolution collision detection, because the aggressiveness of the runtime culling will be limited. The culling efficiency will be maximised 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 multi-resolution representation. In §18.4.4, 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 §18.3.1).

18.6.3. Inherent Limitations of Multi-Resolution 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 multi-resolution 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 has been addressed by Otaduy et al. [OJSL04] and it is discussed in the next section.

19. Perceptually-Driven 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], but the problem of displaying interaction forces and torques between two textured models is mostly unexplored. In fact, computation of texture-induced forces using full-resolution geometric representations of the objects and handling contacts at micro-geometric scale is computationally prohibitive, and novel representations must be considered.

In §18, we have described *contact levels of detail* (CLODs) [OL03b, OL03a], a multi-resolution collision detection algorithm especially designed for 6-DoF haptic rendering that minimises 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 section 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. Otaduy et al. [OJSL04] refer to these texture images as *haptic textures*. This section describes an 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 6-DoF haptic texture 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. Otaduy et al. [OJSL04] synthesised a perceptually inspired force model for haptic texture rendering based on these findings. Force and torque are computed based on the gradient of the directional penetration depth between two textured models. They also introduced 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.

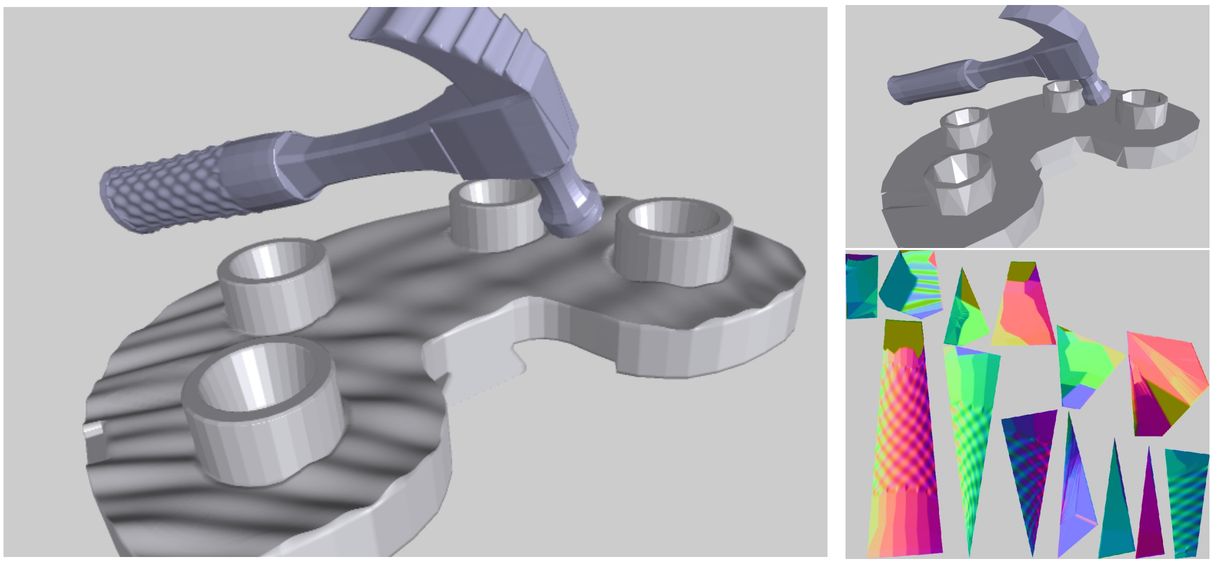


Figure 62: ***Haptic Rendering of Interaction between Textured Models.*** Top: high-resolution textured hammer (433K polygons) and CAD part (658K polygons); Bottom left: low-resolution models (518 & 720 polygons); Bottom right: hammer texture with fine geometric detail.

The 6-DoF haptic texture rendering algorithm has been successfully tested and demonstrated on several complex textured models. One example, consisting of a textured hammer interacting with a rough CAD part, is shown in Figure 62. Subjects that experienced that example were able to perceive roughness induced by surface texture of the objects.

The influence of the perceptual factors identified by psychophysics studies on the vibratory motion induced by the force model has been analysed as well. The experiments demonstrate a qualitative match between roughness perception in earlier experimental observations and the forces simulated using my model. The effectiveness of the rendering algorithm for conveying roughness sensations has been evaluated during both translational and rotational motion. Finally, the performance of the algorithm and its implementation on complex benchmarks has been tested, obtaining force update rates higher than 100Hz.

This section compiles work and results previously published in [OJSL04] and [OL04]. The rest of the section is organised as follows. §19.1 defines the notation used throughout the section and key terminology related to the concept of penetration depth. §19.2 presents the foundations of the rendering algorithm and the force model, which is described in §19.3. §19.4 introduces a simple yet effective algorithm for approximating directional penetration depth and its parallel implementation on graphics processors. Then the experiments and results are described in §19.5, and §19.6 concludes with a discussion on limitations of this work.

19.1. Definitions and Terminology

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

19.1.1. Notations

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

$$\mathbf{p}_{xyz} = g(p_u, p_v) = (\mathbf{u} \ \ \mathbf{v} \ \ \mathbf{n})\,(p_u \ \ p_v \ \ h(p_u, p_v))^T . \tag{15}$$

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:

$$h(p_u, p_v) = \mathbf{n} \cdot g(p_u, p_v). \tag{16}$$

19.1.2. 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 Figure 63-b). As mentioned in §17.1.3, 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 Figure 63-c). The penetration depth between two intersecting surface patches will be referred to as *local penetration depth*.

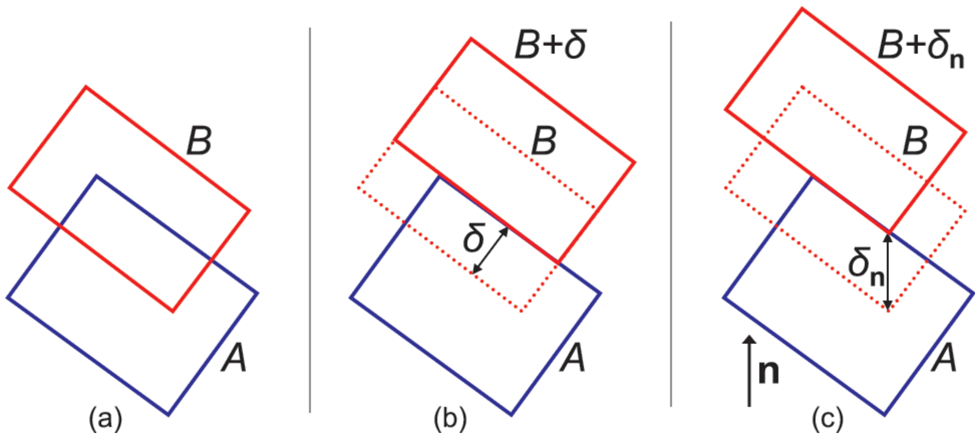


Figure 63: ***Definitions of Penetration Depth.*** (a) Intersecting objects A and B, (b) global penetration depth δ, and (c) directional penetration depth δ_n along $\mathbf{n}$.

Let us assume that two intersecting surface patches S_A and S_B can be represented as height fields along a direction $\mathbf{n}$. Consequently, S_A and S_B can be parameterised by orthographic projection along $\mathbf{n}$, as expressed in §19.1.1. The parameterisation 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 $\mathbf{n}$, as illustrated in Figure 64 by a 2D example.

Therefore, the directional penetration depth $\delta_{\mathbf{n}}$ can be defined as:

$$\delta_{\mathbf{n}} = \max_{(u,v)\in(D_A \cap D_B)} \left(h_A(u,v) - h_B(u,v)\right). \tag{17}$$

19.2. Foundations of a 6-DoF Haptic Texture Rendering Algorithm

In this section we 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 §16.3.2, we have summarised the results of psychophysics studies on perception of roughness that guide the design of the force model. In this section we extend the conclusions from those studies to more general settings and we introduce the rendering pipeline based on haptic textures.

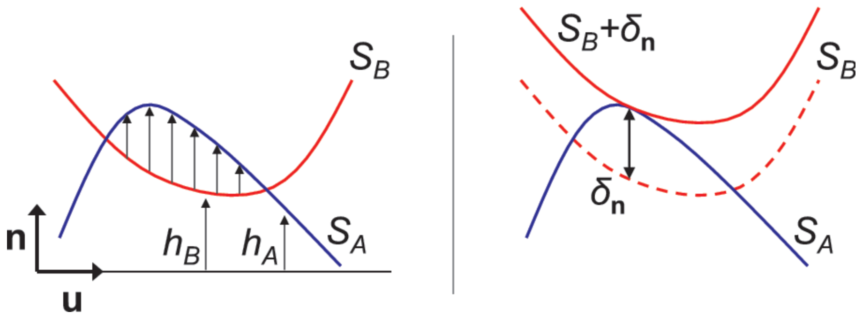


Figure 64: ***Penetration Depth of Height Fields.*** Directional penetration depth of surface patches expressed as height difference.

19.2.1. 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 Figure 65. 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.

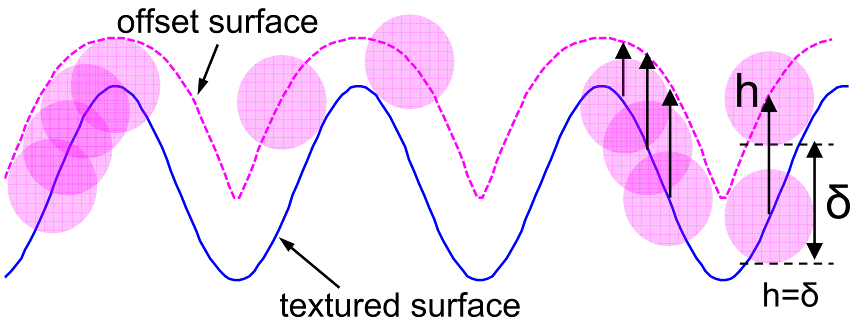


Figure 65: ***Offset Surfaces.*** Left: offset surface computed as the convolution of a surface with a sphere; Centre: sphere whose trajectory traces an offset surface; Right: correspondence between vertical penetration depth (δ) and height of the offset surface (h).

In the design of a force model for haptic texture rendering, one faces the question: How can the concept of offset surface be generalised to the interaction between two arbitrary surfaces? To answer this question, let us consider the case of a spherical probe whose centre moves along a textured surface, as depicted in Figure 65. 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 generalised 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, the force model for general surfaces takes 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 generalisation of the concept used in 3-DoF haptic rendering.

19.2.2. 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.

Following this classification, Otaduy et al. [OJSL04] designed 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, they 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 Characterise 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 section 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 can be achieved integrating the force model in a rendering pipeline such as the one suggested by Otaduy and Lin [OL05].

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 parameterised and, in case of using CLODs, the parameterisation must be consistent across all levels of the hierarchy, and distortion must be minimised. One possible approach is to integrate parameterisation procedures based on existing techniques [SSGH01, COM98] in the *sensation preserving simplification* process for creating CLODs [OL03b]. For the models represented by CLODs, the low-resolution contact information will be obtained following the multi-resolution collision detection algorithm described in §18.

19.3. Perceptually-Driven Force Model

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

19.3.1. 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, Otaduy et al. [OJSL04] adopted 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 modelled explicitly. Nevertheless, Otaduy et al. [OJSL04] analysed the dynamic behaviour 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 §19.5.1.

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.

19.3.2. 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. Otaduy et al. [OJSL04] extended this principle to compute texture-induced forces between two objects. They defined an elastic penetration energy U with stiffness k as:

$$U = \frac{1}{2} k \delta^2. \tag{18}$$

Based on this energy, force $\mathbf{F}$ and torque $\mathbf{T}$ are defined as:

$$\begin{pmatrix} \mathbf{F} \\ \mathbf{T} \end{pmatrix} = -\nabla U = -k\delta \left(\nabla \delta\right), \tag{19}$$

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 §19.2.2, 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 $\mathbf{n}$. One may assume that, locally, the penetration depth between objects A and B can be approximated by the directional penetration depth $\delta_\mathbf{n}$ along $\mathbf{n}$. Then, Eq. 19 is rewritten for $\delta_\mathbf{n}$ in a reference system $\{\mathbf{u}, \mathbf{v}, \mathbf{n}\}$ located at the centre of mass of A. The axes $\mathbf{u}$ and $\mathbf{v}$ may be selected arbitrarily as long as they form an orthonormal basis with $\mathbf{n}$. Eq. 19 reduces to:

$$\begin{pmatrix} F_u \; F_v \; F_n \; T_u \; T_v \; T_n \end{pmatrix}^T = -k\delta_\mathbf{n} \begin{pmatrix} \frac{\partial \delta_\mathbf{n}}{\partial u} \; \frac{\partial \delta_\mathbf{n}}{\partial v} \; 1 \; \frac{\partial \delta_\mathbf{n}}{\partial \theta_u} \; \frac{\partial \delta_\mathbf{n}}{\partial \theta_v} \; \frac{\partial \delta_\mathbf{n}}{\partial \theta_n} \end{pmatrix}^T, \tag{20}$$

where θ_u, θ_v and θ_n are the rotation angles around the axes $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{n}$ 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:

$$\begin{aligned} \mathbf{F}_A &= (\mathbf{u} \;\; \mathbf{v} \;\; \mathbf{n}) \left(F_u \; F_v \; F_n\right)^T, \\ \mathbf{T}_A &= (\mathbf{u} \;\; \mathbf{v} \;\; \mathbf{n}) \left(T_u \; T_v \; T_n\right)^T. \end{aligned} \tag{21}$$

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

Generalising Minsky's approach [Min95], the tangential forces F_u and F_v are proportional to the gradient of penetration depth. However, the force model also defines 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].

19.3.3. Penetration Depth and Gradient

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

$$\frac{\partial\delta_{\mathbf{n}}}{\partial u} = \frac{\delta_{\mathbf{n}}(u+\Delta u, v, n, \theta_u, \theta_v, \theta_n) - \delta_{\mathbf{n}}(u-\Delta u, v, n, \theta_u, \theta_v, \theta_n)}{2\Delta u}, \tag{22}$$

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.

19.4. GPU-Based Approximation of Penetration Depth

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

19.4.1. 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 $f_A : \hat{S}_A \to S_A$, a mapping function from the low-resolution surface patch $\hat{S}_A$ to the surface patch S_A.

As expressed in §19.2.2, 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 §19.1.1. 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 $g_A : D_A \to S_A$ and $\hat{g}_A : \hat{D}_A \to \hat{S}_A$. One can define $\bar{D}_A = D_A \cap \hat{D}_A$.

The mapping function g_A can be approximated by a composite mapping function $f_A \circ \hat{g}_A : \bar{D}_A \to S_A$ (See Figure 66). From Eq. 16, an approximate height function $\hat{h}_A : \bar{D}_A \to \mathbb{R}$ is defined as:

$$\hat{h}_A(u,v) = \mathbf{n}\cdot(f_A \circ \hat{g}_A(u,v)). \tag{23}$$

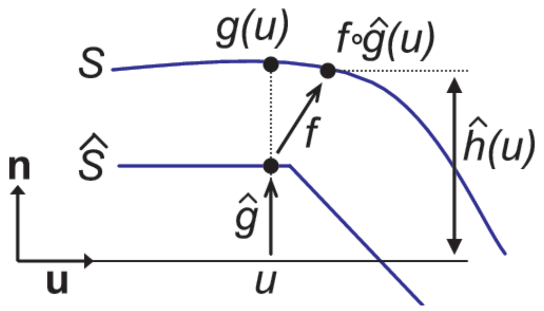


Figure 66: ***Approximate Height Function.*** Height function of a surface patch approximated by a composite mapping function.

Given approximate height functions $\hat{h}_A$ and $\hat{h}_B$, a domain $D = \bar{D}_A \cap \bar{D}_B$, and Eq. 17, the directional penetration depth $\delta_{\mathbf{n}}$ of S_A and S_B can be approximated by:

$$\hat{\delta_{\mathbf{n}}} = \max_{(u,v)\in D}\left(\hat{h}_A(u,v) - \hat{h}_B(u,v)\right). \tag{24}$$

Even though the computation of $\hat{\delta_{\mathbf{n}}}$ can be realised on CPUs, it is best suited for implementation on graphics processors (GPUs), as we will discuss next.

19.4.2. Computation on Graphics Hardware

As shown in Eq. 20, computation of 3D texture-induced force and torque according to the texture force model requires the computation of directional penetration depth $\delta_\mathbf{n}$ and its gradient at every contact. From Eq. 22, this requirement reduces to computing $\delta_\mathbf{n}$ all together at 11 configurations of object A. 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. As pointed out in §17.1.3, 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. 23 and 24 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, first one must 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 $\hat{\delta_\mathbf{n}} = \max(\Delta\hat{h}) = \max\left(\hat{h}_A - \hat{h}_B\right)$. The height difference at the actual object configuration is denoted 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 $\Delta\hat{h}(\pm\Delta\theta_n)$.

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. Otaduy et al. [OL04] 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.

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

Max Search

The *max* function in Eq. 24 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, their difference is computed and stored in the depth buffer. Then the height difference is scaled and offset 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}. A quad is rendered at depth 2^{N-1} and an occlusion query is performed (see http://www.nvidia.com/dev_content/nvopenglspecs/GL_NV_occlusion_query.txt), 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.

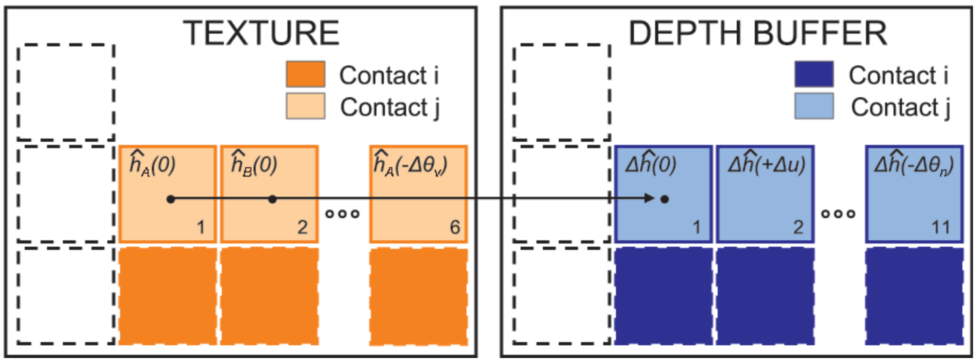


Figure 67: ***Tiling in the GPU.*** Tiling of multiple height functions and contacts to minimise context switches between target buffers.

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 minimise context switches and increase performance (See Figure 67).

Moreover, the domain of each height function is split into 4 quarters, each of which is mapped to one of the RGBA channels. This optimisation exploits vector computation capabilities of fragment processors. As shown in Figure 67, one can 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 optimised. In order to limit context switches, the height functions associated with multiple pairs of contact patches are tiled in one single texture t, and the height differences are tiled in the depth buffer as well, as shown in Figure 67. The cost of *max search* operations is further minimised by performing occlusion queries on all contacts in parallel.

19.5. Experiments and Results

Otaduy et al. [OL04, OJSL04] performed two types of experiments in order to analyse the force model and rendering algorithm for 6-DoF haptic texture rendering. On the one hand, they performed offline experiments to analyse the influence of the factors highlighted by perceptual studies on the vibratory motion induced by the force model [OL04]. On the other hand, they performed interactive experiments to test the effectiveness of the force model and the performance of its implementation [OJSL04].

19.5.1. Comparison with Perceptual Studies

As mentioned in §16.3.2, Klatzky and Lederman conducted experiments where users explored textured plates with spherical probes, and they reported subjective values of perceived roughness. Otaduy and Lin [OL04] created simulated replicas of the physical setups of Klatzky and Lederman's experiments in order to analyse 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 modelled as a circular disk of diameter D and the textured plate as a sinusoidal curve, as shown in Figure 68. The circular disk moves along a horizontal line, which represents a low-resolution approximation of the sinusoidal curve. At each position of the disk the vertical penetration depth $\delta_\mathbf{n}$ with respect to the sinusoidal curve is computed.

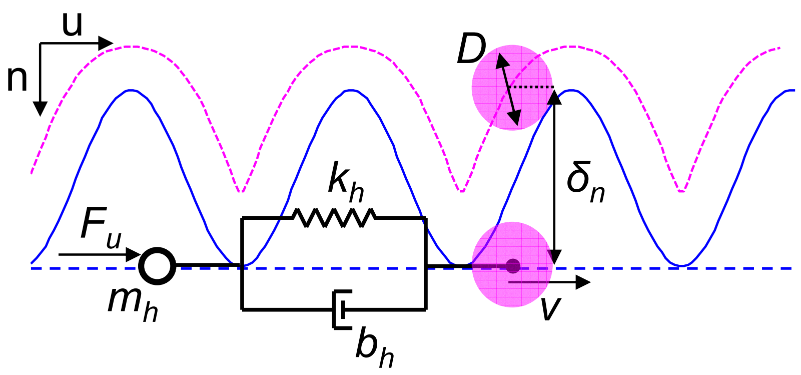


Figure 68: ***Model of Probe-Surface Interaction and Grasping Dynamics.*** A disk moves on a sinusoidal texture at constant speed v while dragging a mass m_h. A texture force F_u, based on penetration depth $\delta_\mathbf{n}$, is applied to the mass.

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

$$F_n = -k\delta_\mathbf{n}, \tag{25}$$

$$F_u = -k\delta_\mathbf{n}\frac{d\delta_\mathbf{n}}{du}. \tag{26}$$

The normal force F_n is one of the factors studied by Lederman et al. [LKHG00]. It is considered as an input in the experiments. Then, one can rewrite:

$$F_u = F_n \frac{d\delta_{\mathbf{n}}}{du}. \tag{27}$$

Human dynamics are modelled 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 analysed by Lederman et al. [LKHR99]. Figure 68 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:

$$m_h \frac{d^2u}{dt^2} = k_h\,(vt - u) + b_h \left(v - \frac{du}{dt} \right) - F_u. \tag{28}$$

The experiments summarised by Klatzky and Lederman [KL02] reflect graphs of perceived roughness vs. texture spacing, both in logarithmic scale. The motion of the hand model has been simulated in Matlab, based on Eq. 28. Subjective roughness values cannot be estimated in the simulations. Instead, knowing that roughness is perceived through vibration, the vibration during simulated interactions is quantified by measuring maximum tangential acceleration values. More specifically, Otaduy and Lin [OL04] measured $max(\frac{d^2u}{dt^2})$ once the motion of the mass reached a periodic state.

Effects of Probe Diameter

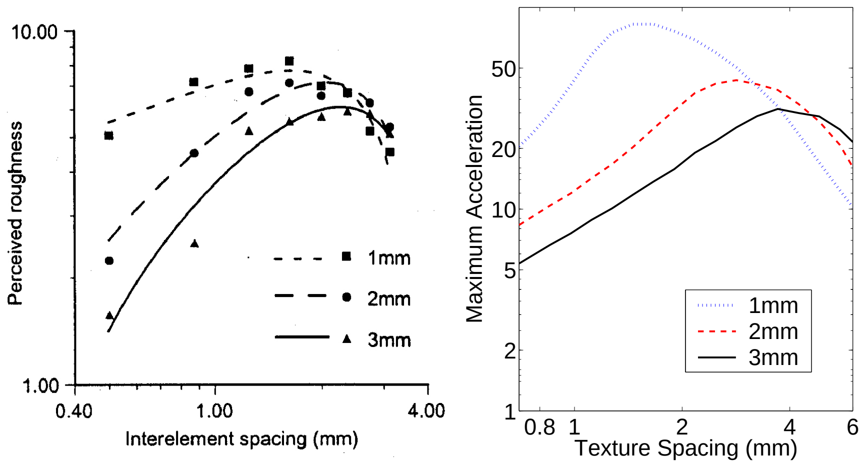


Figure 69: ***Effects of Probe Diameter.*** Left: results of psychophysics studies by Lederman et al. [2000] (printed with permission of ASME and authors); Right: simulation results using a novel force model.

Figure 69 compares 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 Figure 70 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.

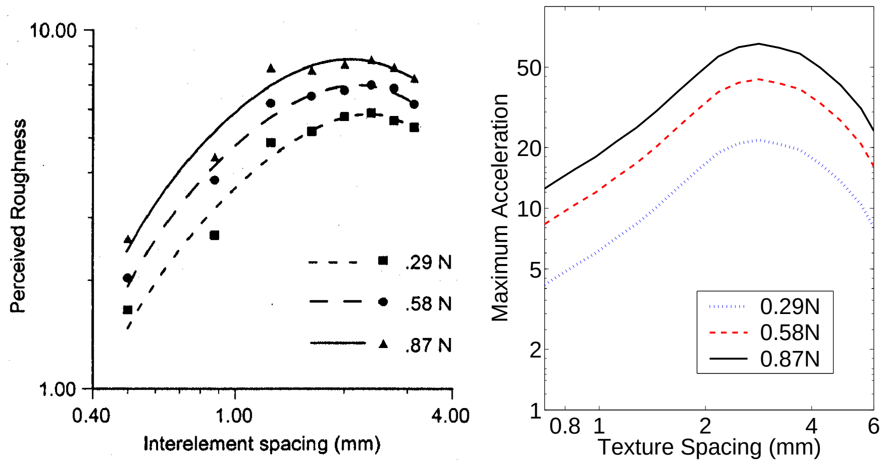


Figure 70: ***Effects of Applied Force.*** Left: results of psychophysics studies by Lederman et al. [2000] (printed with permission of ASME and authors); Right: simulation results using a novel force model.

Effects of Exploratory Speed

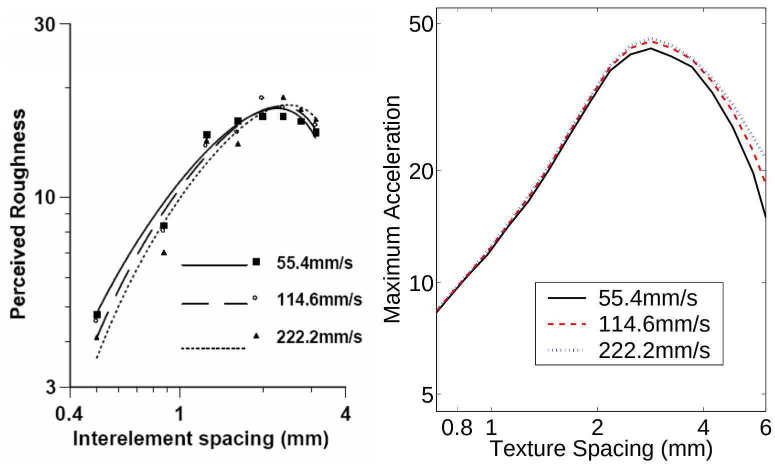


Figure 71: ***Effects of Exploratory Speed.*** Left: results of psychophysics studies by Lederman et al. [1999] (printed with permission of Haptics-e and authors); Right: simulation results using a novel force model.

Figure 71 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 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 §19.3.2 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 was found in the experiments.

19.5.2. Interactive Tests with Complex Models

Otaduy et al. [OJSL04] performed experiments to test the performance of the texture force computation and the rendering algorithm in interactive demonstrations. The first set of experiments evaluated the conveyance of roughness effects under translational and rotational motion. The second set of experiments tested the performance of the haptic texture rendering algorithm and its GPU-based implementation in scenarios with complex contact configurations.

Besides these experiments, several subjects 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 were performed using a 6-DoF PHANTOM 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 was 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 was displayed on a separate commodity PC.

As described in §19.2.2, the first step in the computation of collision response is to find contact information between coarse-resolution models. In a general case, this is done using contact levels of detail (CLODs) for multi-resolution collision detection, as described in §18. 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, a bounding volume hierarchy (BVH) of convex hulls is computed, equivalent to creating CLODs where all levels of the hierarchy are "free" CLODs (see §18.3.3).

Following the approach developed by 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, each value of penetration depth between contact patches is computed 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 stabilising 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. For higher stability, the output of collision response may be integrated in a more stable haptic rendering pipeline such as the one presented in [OL05].

Benchmarks Models and Scenarios

The models shown in Figure 72 were used for the experiments on conveyance of roughness. The performance tests were executed on the models shown in Figures 73 and 74. The complexities of the full-resolution textured models and their coarse resolution approximations are listed in Table 6.

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 analysing 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 Figure 72-b. Figure 75 shows the position of the block and the force exerted on it during $1,500$ frames of interactive simulation (approx. 3 seconds).

Notice that the force in the x direction, which is parallel to the ridges, is almost zero. The texture force model successfully

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 6: ***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).

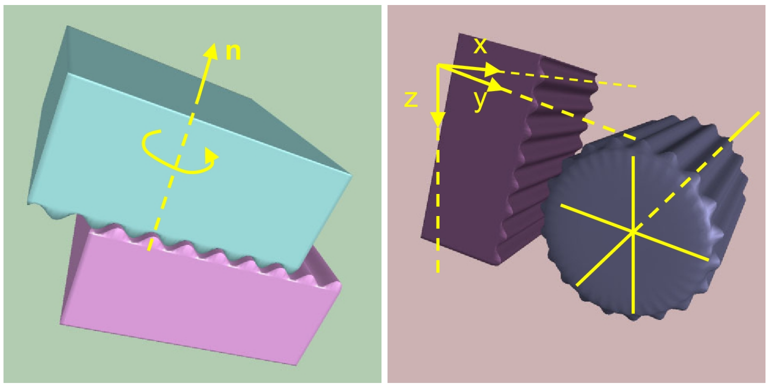


Figure 72: ***Benchmark Models for Experiments on Conveyance of Roughness.*** Left (a): textured blocks; Right (b): block and gear.

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 Figure 72-a. Then small oscillating rotations of the upper block were performed around the direction **n**, and the induced translation along that same direction was observed. Figure 76 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 §19.3 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.

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.

The performance of the haptic texture rendering algorithm and its implementation were also tested 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 Figures 74 and 73).

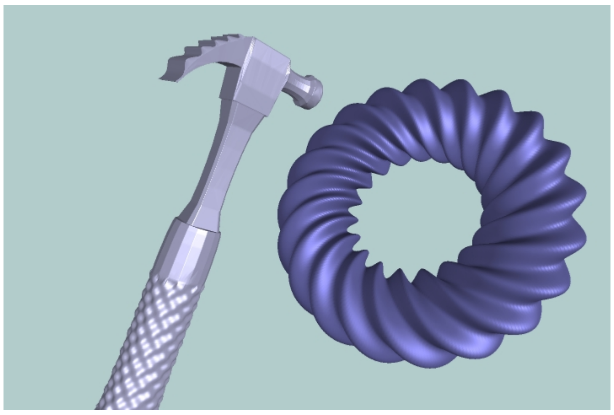


Figure 73: ***Textured Hammer and Helicoidal Torus.*** Benchmark scenario for performance tests.

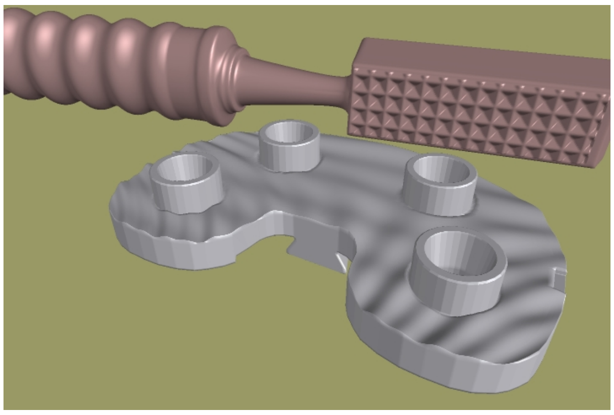


Figure 74: ***File and CAD Part.*** Benchmark scenario for performance tests.

In particular, Figure 77 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 §19.4.2). 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, Figure 77 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.

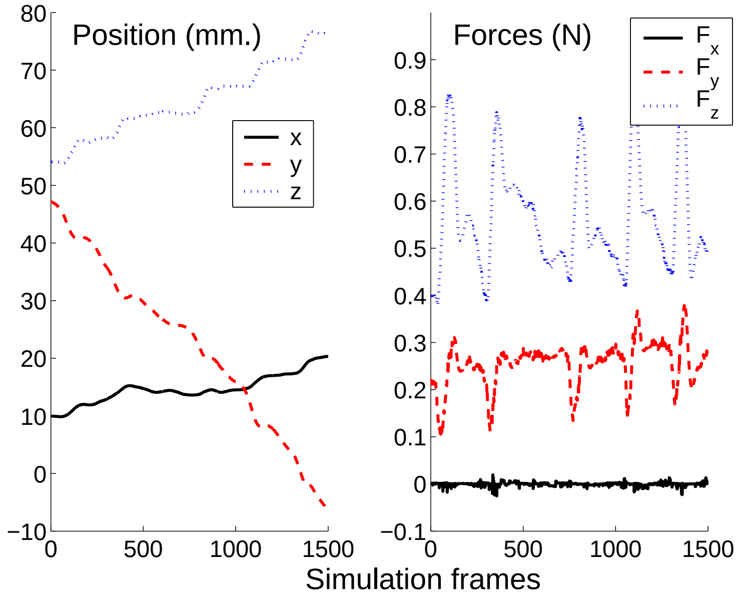


Figure 75: ***Roughness under Translation.*** Position and force profiles generated while translating the model of a textured block in contact with a gear model, as shown in Figure 72-b. Notice the staircase-like motion in *z*, and the correlation between force and position changes.

19.6. Discussion and Limitations

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

19.6.1. 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 §19.5, in practise 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 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 [PC99a] 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.

19.6.2. Frequency and Sampling Issues

As with other sample-based techniques, the haptic texture rendering algorithm is susceptible to aliasing problems. Here we 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 down-sample and reduce their size.

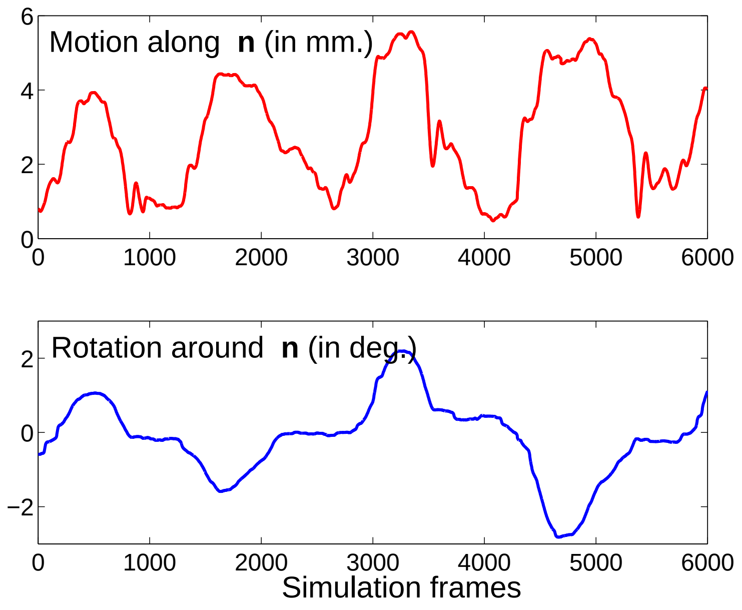


Figure 76: ***Roughness under Rotation.*** Motion profile obtained by rotating one textured block on top of another one, as depicted in Figure 72-a. Notice the translation induced by the interaction of ridges during the rotational motion.

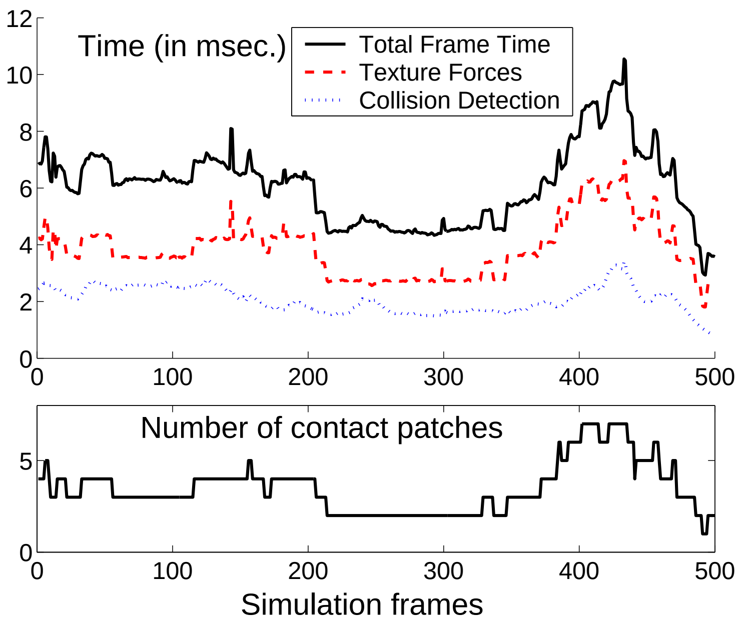


Figure 77: ***Timings.*** Performance analysis and number of clustered contact patches during 500 simulation frames of a file model scraping a CAD part, as shown in Figure 74. In this complex contact scenario the haptic frame rate varies between 100Hz and 200Hz.

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 super-sampling appears to be a solution to the problem, but is often infeasible due to the high update rates required by haptic simulation.

Acknowledgements

We would like to thank Prof. Ming C. Lin from the University of North Carolina at Chapel Hill, who led the research projects presented in §18 and §19. We would also like to thank Stephen Ehmann and Young Kim for their help with SWIFT++ and DEEP; Nitin Jain, Avneesh Sud, and Naga Govindaraju for their contributions in haptic texture rendering; Roberta Klatzky and Susan Lederman for their insights on haptic perception; and Bill Baxter, Mark Foskey, Dinesh Manocha, Jack Snoeyink, Russell Taylor, and Fred Brooks for their valuable feedback and comments.

References

[AB95] AZUMA R. T., BISHOP G.: A frequency domain analysis of head-motion prediction. In *SIGGRAPH Computer Graphics* (1995), pp. 410–408.

[AGHP*00] AGARWAL P., GUIBAS L., HAR-PELED S., RABINOVITCH A., SHARIR M.: Penetration depth of two convex polytopes in 3d. *Nordic J. Computing 7* (2000), 227–240.

[AH98a] ADAMS R. J., HANNAFORD B.: 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] ASTLEY O. R., HAYWARD V.: Multirate haptic simulation achieved by coupling finite element meshes through norton equivalents. *Proc. of IEEE International Conference on Robotics and Automation* (1998).

[AKH01] ADAMS R. J., KLOWDEN D., HANNAFORD B.: Virtual training for a manual assembly task. *Haptics-e 2*, 2 (October 2001).

[AKO95] ADACHI Y., KUMANO T., OGINO K.: Intermediate representation for stiff virtual objects. *Virtual Reality Annual International Symposium* (1995), 203–210.

[And02] ANDRIOT C.: Advances in virtual prototyping. *Clefs CEA Vol. 47, Research and Simulation* (2002).

[AS96] AVILA R. S., SOBIERAJSKI L. M.: A haptic interaction method for volume visualization. In *IEEE Visualization '96* (Oct. 1996), IEEE. ISBN 0-89791-864-9.

[AW00] ARSENAULT R., WARE C.: Eye-hand co-ordination with force feedback. In *SIGCHI Conference on Human Factors in Computing Systems* (The Hague, The Netherlands, 2000), pp. 408–414.

[AWD*04] ADAMS B., WICKE M., DUTRE P., GROSS M., PAULY M., TESCHNER M.: Interactive 3d painting on point-sampled objects. *Eurographics Symposium on Point-Based Graphics* (2004).

[AZWS04] ALLISON R. S., ZACHER J. E., WANG D., SHU J.: Effects of network delay on a collaborative motor task with telehaptic and televisual feedback. In *ACM SIGGRAPH International Conference on Virtual Reality Continuum and its Applications in Industry* (Singapore, 2004), ACM Press, pp. 375–381.

[Bar89] BARAFF D.: Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *Computer Graphics (SIGGRAPH '89 Proceedings)* (July 1989), Lane J., (Ed.), vol. 23, pp. 223–232.

[Bar91] BARAFF D.: Coping with friction for non-penetrating rigid body simulation. In *Computer Graphics (SIGGRAPH '91 Proceedings)* (July 1991), Sederberg T. W., (Ed.), vol. 25, pp. 31–40.

[Bar92] BARAFF D.: *Dynamic simulation of non-penetrating rigid body simulation*. PhD thesis, Cornell University, 1992.

[Bar94] BARAFF D.: Fast contact force computation for nonpenetrating rigid bodies. In *Proceedings of SIGGRAPH '94* (1994), Glassner A., (Ed.), ACM SIGGRAPH, pp. 23–34. ISBN 0-89791-667-0.

[BBFG94] BENFORD S., BOWERS J., FAHLÉN L. E., GREENHALGH C.: Managing mutual awareness in collaborative virtual environments. In *Virtual Reality Software and Technology* (Singapore, 1994), pp. 223–236.

[BC04] BROWN E., CAIRNS P.: A grounded investigation of game immersion. In *Conference on Human Factors in Computing Systems (CHI) extended abstracts on Human factors in computing systems* (Vienna, Austria, 2004), pp. 1297–1300.

[BCN97] BOUVIER E., COHEN E., NAJMAN L.: From crowd simulation to airbag deployment: Particle systems, a new paradigm of simulation. *Electronic Imaging 6*, 1 (1997), 94–107.

[Ber99] BERKELMAN P. J.: *Tool-Based Haptic Interaction with Dynamic Physical Simulations Using Lorentz Magnetic Levitation*. PhD thesis, Carnegie Mellon University, 1999.

[Ber01] BERGEN G. V.: Proximity queries and penetration depth computation on 3d game objects. *Game Developers Conference* (2001).

[BFL*01] BASERMANN A., FINGBERG J., LONSDALE G., MAERTEN B., WALSHAW C.: Dynamic multi-partitioning for parallel finite element applications. *Parallel Computing 27* (2001), 869–881.

[BFM01] B B. W., FRIEDMAN A., MCGAFFEY A.: Measuring and predicting visual fidelity. In *SIGGRAPH* (2001), ACM Press, pp. 213–220.

[BGL97] BENFORD S. D., GREENHALGH C. M., LLOYD D.: Crowded collaborative virtual environments. In *ACM Conference on Human Factors (CHI)* (Atlanta, Georgia, 1997).

[BHM*92] BLAU B., HUGHES C. E., MOSHELL J. M., LISLE C., ORLANDO F.: Networked virtual environments. In *Siggraph Special Issue: Symposium on Interactive 3D Graphics* (Cambridge MA, 1992), pp. 157–164.

[BHS97] BASDOGAN C., HO C.-H., SRINIVASAN M. A.: A ray-based haptic rendering technique for displaying shape and texture of 3d objects in virtual environments. *DSC-Vol. 61, Proceedings of the ASME Dynamic Systems and Control Division* (1997), pp. 77–84.

[BHS01] BASDOGAN C., HO C. H., SRINIVASAN M. A.: Virtual environments for medical training: Graphical and haptic simulation of laparoscopic common bile duct exploration. *IEEE/ASME Transactions on Mechatronics 6*, 3 (September 2001), 269–285.

[BHSS00] BASDOGAN C., HO C.-H., SHRINIVASAN M. A., SLATER M.: An experimental study on the role of touch in shared virtual environments. *ACM Transactions on Computer Human Interaction 7*, 4 (December 2000), 443–460.

[BHW94] BREEN D. E., HOUSE D. H., WOZNY M. J.: Predicting the drape of woven cloth using interacting particles. In *Twenty First Annual Conference on Computer Graphics and Interactive Techniques* (1994), ACM Press, pp. 365–372.

[BKSS90] BECKMANN N., KRIEGEL H., SCHNEIDER R., SEEGER B.: The r*-tree: An efficient and robust access method for points and rectangles. *Proc. SIGMOD Conf. on Management of Data* (1990), 322–331.

[Bli78] BLINN J. F.: Simulation of wrinkled surfaces. In *Computer Graphics (SIGGRAPH '78 Proceedings)* (1978), pp. 286–292.

[Bli82] BLINN J.: Light reflection functions for simulation of clouds and dusty surfaces. *Computer Graphics 16*, 3 (July 1982), 21–29.

[BM98] BOLIN M. R., MEYER G. W.: A perceptually based adaptive sampling algorithm. In *SIGGRAPH Computer Graphics* (July 1998), pp. 299–310.

[BN98] BRO-NIELSEN M.: Finite element modeling in surgery simulation. In *IEEE* (March 1998), vol. 86, pp. 490–503.

[BOH97] BUTTOLO P., OBOE R., HANNAFORD B.: Architectures for shared haptic virtual environments. *Special Issue of Computers and Graphics* (1997).

[BOYBK90] BROOKS, JR. F. P., OUH-YOUNG M., BATTER J. J., KILPATRICK P. J.: Project GROPE — Haptic displays for scientific visualization. In *Computer Graphics (SIGGRAPH '90 Proceedings)* (Aug. 1990), Baskett F., (Ed.), vol. 24, pp. 177–185.

[Bro99] BROOKS JR F. P.: What's real about virtual reality. *IEEE Computer graphics and applications* (1999), 16–27.

[BRS*03] BARANOSKI G. V. G., ROKNE J. G., SHIRLEY P., TRONDSEN T. S., BASTOS R.: Simulating the aurora. *The Journal of Visualization and Computer Animation 14*, 1 (February 2003), 43–59.

[BS80] BEJCZY A., SALISBURY J. K.: Kinematic coupling between operator and remote manipulator. *Advances in Computer Technology Vol. 1* (1980), 197–211.

[BSLM01] BAXTER W., SCHEIB V., LIN M., MANOCHA D.: DAB: Haptic painting with 3D virtual brushes. *Proc. of ACM SIGGRAPH* (2001), 461–468.

[BT89] BERETSEKAS D., TSITSIKLIS J.: *Parallel and Distributed Computation – Numerical Methods*. Prentice Hall, 1989.

[Bur96] BURDEA G. C.: *Force and Touch Feedback for Virtual Reality*. John Wiley and Sons, Inc., 1996.

[But87] BUTCHER J. C.: *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*. Wiley-Interscience, New York, 1987.

[BW97] BLOOMENTHAL J., WYVILL B. (Eds.): *Introduction to Implicit Surfaces*. Barnes & Noble, 1997.

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *SIGGRAPH* (1998), pp. 43–54.

[BW01] BARAFF D., WITKIN A.: *Physically-Based Modeling*. ACM SIGGRAPH Course Notes, 2001.

[BWH03] BEEHAREE A. K., WEST A. J., HUBBOLD R.: Visual attention based information culling for distributed virtual environments. In *Tenth ACM Symposium on Virtual Reality Software and Technology (VRST)* (October 2003), ACM Press, pp. 213–222.

[BWK03] BARAFF D., WITKIN A., KASS M.: Untangling cloth. In *SIGGRAPH* (July 2003), vol. 22, ACM Press, pp. 862–870.

[Cam97] CAMERON S.: Enhancing GJK: Computing minimum and penetration distance between convex polyhedra. *IEEE International Conference on Robotics and Automation* (1997), 3112–3117.

[Cat74] CATMULL E. E.: *A Subdivision Algorithm for Computer Display of Curved Surfaces*. PhD thesis, Dept. of CS, U. of Utah, Dec. 1974.

[CB94] COLGATE J. E., BROWN J. M.: Factors affecting the z-width of a haptic display. *IEEE International Conference on Robotics and Automation* (1994), 3205–3210.

[CC86] CAMERON S., CULLEY R. K.: Determining the minimum translational distance between two convex polyhedra. *Proceedings of International Conference on Robotics and Automation* (1986), 591–596.

[CC97] CHANG B., COLGATE J. E.: Real-time impulse-based simulation of rigid body systems for haptic display. *Proc. of ASME Dynamic Systems and Control Division* (1997).

[CCL02] CATER K., CHALMERS A. G., LEDDA P.: Selective quality rendering by exploiting human inattentional blindness: Looking but not seeing. In *Symposium on Virtual Reality Software and Technology* (2002), ACM, pp. 17–24.

[CCW03] CATER K., CHALMERS A., WARD G.: Detail to attention: Exploiting visual tasks for selective rendering. In *Eurographics Symposium on Rendering* (Leuven, Belgium, 2003), Eurographics Association, pp. 270–280.

[CD68] COTTLE R. W., DANTZIG G. B.: Complementarity pivot theory of mathematical programming. *Linear Algebra and its Applications* (1968), 103–125.

[CDA99] COTIN S., DELINGETTE H., AYACHE N.: Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions on Visualization and Computer Graphics 5*, 1 (1999), 62–73.

[CDK01] COULOURIS G., DOLLIMORE J., KINDBERG T.: *Distributed Systems Concepts and Design*, third ed. Addison Wesley, 2001.

[CDR02] CHALMERS A., DAVIS T., REINHARD E.: *Practical Parallel Rendering*. AK Peters, Nantick, USA, 2002.

[CDST97] CHAZELLE B., DOBKIN D., SHOURABOURA N., TAL A.: Strategies for polyhedral surface decomposition: An experimental study. *Computational Geometry: Theory and Applications 7* (1997), 327–342.

[CGD97] CANI-GASCUEL M.-P., DESBRUN M.: Animation of deformable models using implicit surfaces. *IEEE Transactions on Visualisation and Computer Graphics 3*, 1 (March 1997), 39–50.

[CGSS93] COLGATE J. E., GRAFING P. E., STANLEY M. C., SCHENKEL G.: Implementation of stiff virtual walls in force-reflecting interfaces. *Virtual Reality Annual International Symposium* (1993), 202–207.

[Che99] CHEN E.: Six degree-of-freedom haptic system for desktop virtual prototyping applications. In *Proceedings of the First International Workshop on Virtual Reality and Prototyping* (1999), pp. 97–106.

[CJ92] CONNOR C. E., JOHNSON K. O.: Neural coding of tactile texture: Comparison of spatial and temporal mechanisms for roughness perception. *Journal of Neuroscience 12* (1992), pp. 3414–3426.

[CKM03] CHALMERS A. G., K K. C., MAFLIOLI D.: Visual attention models for producing high fidelity graphics efficiently. In *Spring Conference on Computer Graphics* (Budermice, April 2003).

[COM98] COHEN J., OLANO M., MANOCHA D.: Appearance preserving simplification. In *Proc. of ACM SIGGRAPH* (1998), pp. 115–122.

[Cos00] COSTA M. A.: *Fractal Description of Rough Surfaces for Haptic Display*. PhD thesis, Stanford University, 2000.

[CP01] COOK J., PETTIFER S.: Placeworld: An integration of shared virtual worlds. In *ACM SIGGRAPH Sketches and applications* (August 2001), ACM Press, p. 243.

[CPGB94] CONWAY M., PAUSCH R., GOSSWEILER R., BURNETTE T.: Alice: A rapid prototyping system for building virtual environments. In *ACM CHI Human Factors in Computing Systems* (Boston, Massachusetts, 1994), pp. 295–296.

[CPS92] COTTLE R. W., PANG J. S., STONE R. E.: The linear complementarity problem. *Academic-Press, Inc.* (1992).

[CS94] COLGATE J. E., SCHENKEL G. G.: Passivity of a class of sampled-data systems: Application to haptic interfaces. *Proc. of American Control Conference* (1994).

[CSB95] COLGATE J. E., STANLEY M. C., BROWN J. M.: Issues in the haptic display of tool use. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems* (1995), pp. 140–145.

[CSC04] CONSTANTINESCU D., SALCUDEAN S. E., CROFT E. A.: Impulsive forces for haptic rendering of rigid contact. *Proc. of International Symposium on Robotics* (2004), 1–6.

[CT00] CAVUSOGLU M. C., TENDICK F.: Multirate simulation for high fidelity haptic interaction with deformable objects in virtual environments. In *IEEE International Conference on Robotics and Automation* (2000), pp. 2458–2465.

[CT03a] CHOI S., TAN H. Z.: Aliveness: Perceived instability from a passive haptic texture rendering system. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems* (2003).

[CT03b] CHOI S., TAN H. Z.: An experimental study of perceived instability during haptic texture rendering: Effects of collision detection algorithm. *Proc. of Haptics Symposium* (2003), 197–204.

[CTS02] CAVUSOGLU M. C., TENDICK F., SASTRY S. S.: Haptic interfaces to real and virtual surgical environments. In *Touch in Virtual Environments*, McLaughlin M. L., Hespanha J. P.,, Sukhatme G. S., (Eds.). Prentice Hall PTR, Upper Saddle River, NJ, 2002, ch. 13, pp. 217–237.

[CWH93] COHEN M. F., WALLACE J. R., HANRAHAN P.: *Radiosity and Realistic Image Synthesis*. Academic Press Professional, San Diego, CA, USA, 1993.

[Dah99] DAHMANN J. S.: The high level architecture and beyond: Technology challenges. In *Thirteenth Workshop on Parallel and Distributed Simulation* (Atlanta, Georgia, United States, 1999), IEEE Computer Society, pp. 64–70.

[DAK04] DURIEZ C., ANDRIOT C., KHEDDAR A.: A multi-threaded approach for deformable/rigid contacts with haptic feedback. *Proc. of Haptics Symposium* (2004).

[Dal93] DALY S.: The visible differences predictor: An algorithm for the assessment of image fidelity. In *Digital Image and Human Vision* (Cambridge, MA, 1993), Watson A. B., (Ed.), MIT Press, pp. 179–206.

[DCA99] DELINGETTE H., COTIN S., AYACHE N.: A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation. In *Computer Animation* (May 1999), pp. 70–81.

[DDCB01] DEBUNNE G., DESBRUN M., CANI M. P., BARR A. H.: Dynamic real-time deformations using space and time adaptive sampling. *Proc. of ACM SIGGRAPH* (2001).

[DG95] DESBRUN M., GASCUEL M.-P.: Animating soft substances with implicit surfaces. In *Twenty Second Annual Conference on Computer Graphics and Interactive Techniques* (1995), ACM Press, pp. 287–290.

[DH00] DOUGLAS Y., HARGADON A.: The pleasure principle: Immersion engagement and flow. In *Proceedings of Hypertext* (2000), ACM Press, pp. 153–160.

[DHKS93] DOBKIN D., HERSHBERGER J., KIRKPATRICK D., SURI S.: Computing the intersection-depth of polyhedra. *Algorithmica 9* (1993), 518–533.

[DK90] DOBKIN D. P., KIRKPATRICK D. G.: Determining the separation of preprocessed polyhedra – a unified approach. In *Proc. 17th Internat. Colloq. Automata Lang. Program.* (1990), vol. 443 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 400–413.

[DQ*99] DACHILLE F., QIN H., , KAUFMAN A., EL-SANA J.: Haptic sculpting of dynamic surfaces. *Proc. of ACM Symposium on Interactive 3D Graphics* (1999), 103–110.

[DWS*99] DINH H. Q., WALKER N., SONG C., KOBAYASHI A., HODGES L. F.: Evaluating the importance of multisensory input on memory and the sense of presence in virtual environments. In *IEEE Virtual Reality* (1999), pp. 222–228.

[EB01] ERNST M. O., BANKS M. S.: Does vision always dominate haptics? *Touch in Virtual Environments Conference* (2001).

[EDD*95] ECK M., DEROSE T., DUCHAMP T., HOPPE H., LOUNSBERY M., STUETZLE W.: Multiresolution analysis of arbitrary meshes. In *SIGGRAPH 95 Conference Proceedings* (Aug. 1995), Cook R., (Ed.), Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 173–182. held in Los Angeles, California, 06-11 August 1995.

[EHS*97] EDMOND C., HESKAMP D., SLUIS D., STREDNEY D., WIET G., YAGEL R., WEGHORST S., OPPENHEIMER P., MILLER J., LEVIN M., ROSENBERG L.: Ent endoscopic surgical simulator. *Proc. of Medicine Meets VR* (1997), 518–528.

[EL00] EHMANN S., LIN M. C.: Accelerated proximity queries between convex polyhedra using multi-level voronoi marching. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems* (2000), 2101–2106.

[EL01] EHMANN S., LIN M. C.: Accurate and fast proximity queries between polyhedra using convex surface decomposition. *Computer Graphics Forum (Proc. of Eurographics'2001) 20*, 3 (2001), 500–510.

[EM63] EMLING J. W., MITCHELL D.: The effects of time delay and echos on telephone conversations. *Bell System Technical Journal 22* (November 1963), 2869–2891.

[EMF02] ENRIGHT D., MARSCHNER S., FEDKIW R.: Animation and rendering of complex water surfaces. *ACM Transactions on Graphics 21*, 3 (July 2002), 736–744.

[EMTTT98] EARNSHAW R., MAGNENAT-THALMANN N., TERZOPOULOS D., THALMANN D.: Guest editors' introduction: Computer animation for virtual humans. *IEEE Computer Graphics and Applications 18*, 5 (September–October 1998), 20–23.

[EPMW99] ECONOMOU D., PETTIFER S. R., MITCHELL W. L., WEST A. J.: The Kahun project: CVE technology development based on real world application and user needs. In *ACM Symposium on Virtual Reality Software and Technology* (1999), pp. 168–169.

[ESJ97] ELLIS R. E., SARKAR N., JENKINS M. A.: Numerical methods for the force reflection of contact. *ASME Transactions on Dynamic Systems, Modeling and Control 119* (1997), 768–774.

[ESV00] EL-SANA J., VARSHNEY A.: Continuously-adaptive haptic rendering. *Virtual Environments 2000* (2000), pp. 135–144.

[Fer66] FERRELL W. R.: Delayed force feedback. *Human Factors 8* (1966), 449–455.

[FFM*04] FISHER B., FELS S., MACLEAN K., MUNZNER T., RENSINK R.: Seeing, hearing and touching: Putting it all together. In *ACM SIGGRAPH course notes* (2004).

[Fit54] FITTS P. M.: The information capacity of the human motor system in controlling the amplitude of movement. *Experimental Psychology 47*, 6 (1954), 381–391.

[FL01] FISHER S., LIN M. C.: Fast penetration depth estimation for elastic bodies using deformed distance fields. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems 2001* (2001).

[FOL02] FOSKEY M., OTADUY M. A., LIN M. C.: ArtNova: Touch-enabled 3D model design. *Proc. of IEEE Virtual Reality Conference* (2002).

[FPSG96] FERWERDA J., PATTANAIK S. N., SHIRLEY P., GREEENBERG D. P.: A model of visual adaptation for realistic image synthesis. In *SIGGRAPH* (1996), ACM Press, pp. 249–258.

[FR86] FOURNIER A., REEVES W. T.: A simple model of ocean waves. In *ACM SIGGRAPH Computer Graphics* (August 1986), ACM Press, pp. 75–84.

[FS98] FRÉCON E., STENIUS M.: DIVE: A scaleable network architecture for distributed virtual environments. *Distributed Systems Engineering – Special Issue on Distributed Virtual Environments 5*, 3 (September 1998), 91–100.

[FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual simulation of smoke. In *SIGGRAPH Computer Graphics* (2001), ACM Press, pp. 15–22.

[Fuj99] FUJIMOTO R. M.: *Parallel and Distributed Simulation Systems*. Wiley-Interscience, 1999.

[GB95] GREENHALGH C., BENFORD S.: MASSIVE: A collaborative virtual environment for teleconferencing. *ACM Transactions on Computer-Human Interaction (TOCHI) 2*, 3 (September 1995), 239–261.

[GBF03] GUENDELMAN E., BRIDSON R., FEDKIW R.: Nonconvex rigid bodies with stacking. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH) 22* (2003), 871–878.

[GCHH03] GIBSON S., COOK J., HOWARD T. L. J., HUBBOLD R. J.: Rapid shadow generation in real-world lighting environments. In *Eurographics Symposium on Rendering* (Leuven, Belgium, June 2003).

[GEL00] GREGORY A., EHMANN S., LIN M. C.: *inTouch*: Interactive multiresolution modeling and 3d painting with a haptic interface. *Proc. of IEEE VR Conference* (2000).

[GFPB02] GREENHALGH C., FLINTHAM M., PURBRICK J., BENFORD S.: Applications of temporal links: Recording and replaying virtual environments. In *IEEE Virtual Reality Conference* (Orlando, Florida, March 2002), pp. 101–108.

[GH97a] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *Proc. of ACM SIGGRAPH* (1997), pp. 209–216.

[GH97b] GIBSON S., HUBBOLD R. J.: Perceptually-driven radiosity. *Computer Graphics Forum 16*, 2 (June 1997), 129–141.

[GHAH03] GUNN C., HUTCHINS M., ADCOCK M., HAWKINS R.: Trans-world haptic collaboration. In *SIGGRAPH: Sketches and Applications* (2003).

[GHL05] GLENCROSS M., HUBBOLD R., LYONS B.: Dynamic primitive caching for haptic rendering of large-scale models. In *IEEE worldHAPTICS First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems* (Pisa, Italy, March 2005), pp. 517–518.

[GHP01] GLENCROSS M., HOWARD T., PETTIFER S.: Iota: An approach to physically-based modelling in virtual environments. In *IEEE Virtual Reality Conference* (Yokohama, Japan, March 2001), pp. 287–288.

[GHSA05] GUNN C., HUTCHINS M., STEVENSON D., ADCOCK M.: Using collaborative haptics in remote surgical training. In *worldHAPTICS First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems* (Pisa, Italy, March 2005), IEEE Computer Society, pp. 481–482.

[GHZ99] GUIBAS L., HSU D., ZHANG L.: *H-Walk*: Hierarchical distance computation for moving convex bodies. *Proc. of ACM Symposium on Computational Geometry* (1999).

[GJK88] GILBERT E. G., JOHNSON D. W., KEERTHI S. S.: A fast procedure for computing the distance between objects in three-dimensional space. *IEEE J. Robotics and Automation vol RA-4* (1988), 193–203.

[Gla89] GLASSNER A. S.: *An Introduction to Ray Tracing*. Academic Press, London, UK, 1989.

[Gle00] GLENCROSS M.: *A Framework for Physically Based Modelling in Virtual Environments*. PhD thesis, The University of Manchester, 2000.

[GLGT99] GREGORY A., LIN M., GOTTSCHALK S., TAYLOR R.: H-COLLIDE: A framework for fast and accurate collision detection for haptic interaction. In *Proceedings of Virtual Reality Conference 1999* (1999), pp. 38–45.

[GLM96] GOTTSCHALK S., LIN M., MANOCHA D.: OBB-Tree: A hierarchical structure for rapid interference detection. *Proc. of ACM Siggraph'96* (1996), 171–180.

[GLW*04] GOVINDARAJU N., LLOYD B., WANG W., LIN M., MANOCHA D.: Fast computation of database operations using graphics processors. *Proc. of ACM SIGMOD* (2004).

[GM98] GLENCROSS M., MURTA A.: Multi-body simulation in virtual environments. In *Simulation – Past, Present and Future. Twelveth European Simulation Multiconference* (Manchester, England, June 1998), Zobel R., Moeller D., (Eds.), pp. 590–594.

[GM99] GLENCROSS M., MURTA A.: A virtual jacob's ladder. In *Graphicon* (Moscow, August 1999), pp. 88–94.

[GM00] GLENCROSS M., MURTA A.: Managing the complexity of physically based modelling in virtual reality. In *Fourth International Conference of Computer Graphics and Artificial Intelligence* (Limoges, May 2000), pp. 41–48.

[GME*00] GREGORY A., MASCARENHAS A., EHMANN S., LIN M. C., MANOCHA D.: 6-DoF haptic display of polygonal models. *Proc. of IEEE Visualization Conference* (2000).

[Got00] GOTTSCHALK S.: *Collision Queries using Oriented Bounding Boxes*. PhD thesis, University of North Carolina. Department of Computer Science, 2000.

[GRLM03] GOVINDARAJU N., REDON S., LIN M., MANOCHA D.: CULLIDE: Interactive collision detection between complex models in large environments using graphics hardware. *Proc. of ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware* (2003), 25–32.

[GSM*97] GIBSON S., SAMOSKY J., MOR A., FYOCK C., GRIMSON E., KANADE T.: 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)* (1997), 368–378.

[GSS99] GUSKOV I., SWELDENS W., SCHRODER P.: Multiresolution signal processing for meshes. *Proc. of ACM SIGGRAPH* (1999), pp. 325 – 334.

[GSW97] GUPTA R., SHERIDAN T., WHITNEY D.: Experiments using multimodal virtual environments in design for assembly analysis. *Presense: Teleoperators and Virtual Environments 6*, 3 (1997), 318–338.

[GT54] GOERTZ R., THOMPSON R.: Electronically controlled manipulator. *Nucleonics* (1954), 46–47.

[GTS*97] GREENBERG D. P., TORRANCE K. E., SHIRLEY P., ARVO J., FERWERDA J., PATTANAIK S. N., LAFORTUNE A. E., WALTER B., FOO S.-C., TRUMBORE B.: A framework for realistic image synthesis. In *SIGGRAPH: Special Session* (1997), ACM Press, pp. 477–494.

[Gut01] GUTWIN C.: The effects of network delays on group work in real-time groupware. In *European Conference on CSCW* (Bonn, 2001), pp. 299–318.

[HA00] HAYWARD V., ARMSTRONG B.: A new computational model of friction applied to haptic rendering. *Experimental Robotics VI* (2000).

[Hag96] HAGSAND O.: Interactive multiuser VEs in the DIVE system. *IEEE Multimedia 3*, 1 (1996), 30–39.

[Han97] HAND C.: A survey of 3d interaction techniques. *Computer Graphics Forum 16*, 5 (December 1997), 269–281.

[Har04] HARRIS M.: Fast fluid dynamics simulation on the GPU. In *GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics*, Fernando R., (Ed.). Addison-Wesley, 2004, pp. 637–665.

[HBS99] HO C.-H., BASDOGAN C., SRINIVASAN M. A.: Efficient point-based rendering techniques for haptic display of virtual objects. *Presence 8*, 5 (1999), pp. 477–491.

[HBSL03] HARRIS M. J., BAXTER W. V., SCHEUERMANN T., LASTRA A.: Simulation of cloud dynamics on graphics hardware. In *SIGGRAPH/Eurographics Workshop On Graphics Hardware* (San Diego, California, 2003), ACM Press, pp. 91–101.

[HC02] HASSER C. J., CUTKOSKY M. R.: System identification of the human hand grasping a haptic knob. *Proc. of Haptics Symposium* (2002), 180–189.

[HCGB99] HELLER M. A., CALCATERRA J. A., GREEN S. L., BROWN L.: Intersensory conflict between vision and touch: The response modality dominates when precise, attention-riveting judgements are required. *Perception and Psychophysics 61* (1999), pp. 1384–1398.

[HCSL02] HARRIS M. J., COOMBE G., SCHEUERMANN T., LASTRA A.: Physically-based visual simulation on graphics hardware. In *SIGGRAPH/Eurographics Workshop On Graphics Hardware* (2002), ACM Press.

[Hec92] HECKBERT P. S.: *Simulating Global Illumination Using Adaptive Meshing*. PhD thesis, University of California at Berkeley, 1992.

[HGA*98] HAYWARD V., GREGORIO P., ASTLEY O., GREENISH S., DOYON M.: Freedom-7: A high fidelity seven axis haptic device with applications to surgical training. *Experimental Robotics* (1998), 445–456. Lecture Notes in Control and Information Sciences 232.

[HGMR98] HASSER C. J., GLODENBURG A. S., MARTIN K. M., ROSENBURG L. B.: User performing a GUI pointing task with a low-cost force-feedback computer mouse. In *ASME Dynamic Systems and Control Division* (1998), vol. 64, pp. 151–156.

[HL01] HARRIS M. J., LASTRA A.: Real-time cloud rendering. In *Eurographics* (2001), Blackwells, pp. 76–84.

[HMG03] HUANG G., METAXAS D., GOVINDARAJ M.: Feel the "fabric": An audio-haptic interface. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), pp. 52–61.

[HMS*00] HESPANHA J. P., MCLAUGHLIN M., SUKHATME G. S., AKBARIAN M., GARG R., ZHU W.: Haptic collaboration over the internet. In *The Fifth PHANTOM Users Group Workshop* (2000).

[Hog85] HOGAN N.: Impedance control: An approach to manipulation, part i - theory, part ii - implementation, part iii - applications. *Journal of Dynamic Systems, Measurement and Control 107* (1985), 1–24.

[Hog86] HOGAN N.: Multivariable mechanics of the neuromuscular system. *IEEE Annual Conference of the Engineering in Medicine and Biology Society* (1986), 594–598.

[Hop96] HOPPE H.: Progressive meshes. In *SIGGRAPH 96 Conference Proceedings* (Aug. 1996), Rushmeier H., (Ed.), Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 99–108. held in New Orleans, Louisiana, 04-09 August 1996.

[Hop97] HOPPE H.: View dependent refinement of progressive meshes. In *ACM SIGGRAPH Conference Proceedings* (1997), pp. 189–198.

[HR00] HOLLINS M., RISNER S.: Evidence for the duplex theory of tactile texture perception. *Perception & Psychophysics 62* (2000), 695–705.

[HRF03] HALLORAN J., ROGERS Y., FITZPATRICK G.: From text to talk: Multiplayer games and voiceover IP. In *Level Up: First International Digital Games Research Conference* (2003), pp. 130–142.

[HRK02] HANNAFORD B., RYU J.-H., KIM Y. S.: Stable control of haptics. In *Touch in Virtual Environments*, McLaughlin M. L., Hespanha J. P.„ Sukhatme G. S., (Eds.). Prentice Hall PTR, Upper Saddle River, NJ, 2002, ch. 3, pp. 47–70.

[HS77] HILL J. W., SALISBURY J. K.: Two measures of performance in a peg-in-hole manipulation task with force feedback. *Thirteenth Annual Conference on Manual Control, MIT* (1977).

[HS04] HASEGAWA S., SATO M.: Real-time rigid body simulation for haptic interactions based on contact volume of polygonal objects. In *Proc. of Eurographics* (2004).

[Hub94] HUBBARD P.: *Collision Detection for Interactive Graphics Applications*. PhD thesis, Brown University, 1994.

[Hub02] HUBBOLD R. J.: Collaborative stretcher carrying: A case study. In *ACM Workshop on Virtual Environments* (Barcelona, Spain, 2002), pp. 7–12.

[HZLM01] HOFF K., ZAFERAKIS A., LIN M., MANOCHA D.: Fast and simple 2d geometric proximity queries using graphics hardware. *Proc. of ACM Symposium on Interactive 3D Graphics* (2001), 145–148.

[IK00] ITTI L., KOCH C.: A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research 40*, 10–12 (2000), 1489–1506.

[IKN98] ITTI L., KOCH C., NIEBUR E.: A model of saliency-based visual attention for rapid scene analysis. *Pattern Analysis and Machine Intelligence 20* (1998), 1254–1259.

[IMWB01] INSKO B., MEEHAN M., WHITTON M., BROOKS F.: *Passive Haptics Significantly Enhances Virtual Environments*. Tech. Rep. 01-010, Department of Computer Science, UNC Chapel Hill, 2001.

[Ins01] INSKO B.: *Passive Haptics Significantly Enhance Virtual Environments*. PhD thesis, University of North Carolina. Department of Computer Science, 2001.

[Jam90] JAMES W.: *Principles of Pshychology*. Holt, New York, 1890.

[JC01] JOHNSON D. E., COHEN E.: Spatialized normal cone hierarchies. *Proc. of ACM Symposium on Interactive 3D Graphics* (2001), pp. 129–134.

[Jef85] JEFFERSON D. R.: Virtual time. *ACM Transactions on Programming Languages and Systems 7*, 3 (1985), 404–425.

[Jen01] JENSEN H. W.: *Realistic Image Synthesis Using Photon Mapping*. AK Peters, Nantick, USA, 2001.

[JH04] JAY C., HUBBOLD R.: Does performance in the face of delayed sensory feedback change with practice? In *Eurohaptics* (2004), Springer-Verlag, pp. 530–533. ISBN 3-9809614-1-9.

[JH05] JAY C., HUBBOLD R.: Delayed visual and haptic feedback in a reciprocal tapping task. In *IEEE worldHAPTICS First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems* (Pisa, Italy, March 2005), pp. 655–656. ISBN 0-7695-2310-2/05.

[Jor05] Jorvik: The Viking city. Online document, May 2005. http://www.jorvik-viking-centre.co.uk/jorvik-navigation.htm.

[JP04] JAMES D. L., PAI D. K.: Bd-tree: Output-sensitive collision detection for reduced deformable models. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH)* (2004).

[JTK*99] JOHNSON D., THOMPSON II T. V., KAPLAN M., NELSON D., COHEN E.: Painting textures with a haptic interface. *Proceedings of IEEE Virtual Reality Conference* (1999).

[JW03] JOHNSON D. E., WILLEMSEN P.: Six degree of freedom haptic rendering of complex polygonal models. In *Proc. of Haptics Symposium* (2003).

[JW04] JOHNSON D. E., WILLEMSEN P.: Accelerated haptic rendering of polygonal models through local descent. *Proc. of Haptics Symposium* (2004).

[Kar85] KARNOPP D.: Computer simulation of stick slip friction in mechanical dynamic systems. *Trans. ASME, Journal of Dynamic Systems, Measurement, and Control* (1985).

[Kat89] KATZ D.: *The World of Touch*. Erlbaum, Hillsdale, NJ, 1989. L. Krueger, Trans. (Original work published 1925).

[KB91] KIM W., BEJCZY A.: Graphical displays for operator aid in telemanipulation. *IEEE International Conference on Systems, Man and Cybernetics* (1991).

[KH97] KHOTE M., HOWARD T.: Mirages. In *Eurographics UK Fifteenth Annual Conference* (Norwich, England, March 1997), pp. 223–235.

[KHM*98] KLOSOWSKI J., HELD M., MITCHELL J., SOWIZRAL H., ZIKAN K.: Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Trans. on Visualization and Computer Graphics 4*, 1 (1998), 21–37.

[Kil76] KILPATRICK P. J.: *The use of a kinesthetic supplement in an interactive graphics system*. Ph.d. thesis, The University of North Carolina at Chapel Hill, 1976.

[KKH*97] KUHNAPFEL U. G., KUHN C., HUBNER M., KRUMM H.-G., MAASS H., NEISIUS B.: The karlsruhe endoscopic surgery trainer as an example for virtual reality in medical education. *Minimally Invasive Therapy and Allied Technologies Vol. 6* (1997), 122–125.

[KKT*04] KIM J., KIM H., TAY B. K., MUNIYANDI M., SRINIVASAN M. A., JORDAN J., MORTENSEN J., OLIVEIRA M., SLATER M.: Transatlantic touch: A study of haptic collaboration over long distance. *Presence: Teleoperator and Virtual Environments 13*, 3 (2004), 328–337.

[KL95] KLATZKY R. L., LEDERMAN S. J.: Identifying objects from a haptic glance. *Perception and Psychophysics 57* (1995), pp. 1111–1123.

[KL99] KLATZKY R. L., LEDERMAN S. J.: Tactile roughness perception with a rigid link interposed between skin and surface. *Perception and Psychophysics 61* (1999), pp. 591–607.

[KL02] KLATZKY R. L., LEDERMAN S. J.: Perceiving texture through a probe. In *Touch in Virtual Environments*, McLaughlin M. L., Hespanha J. P.,, Sukhatme G. S., (Eds.). Prentice Hall PTR, Upper Saddle River, NJ, 2002, ch. 10, pp. 180–193.

[KL03] KLATZKY R. L., LEDERMAN S. J.: Touch. In *Experimental Psychology* (2003), pp. 147–176. Volume 4 in I.B. Weiner (Editor-in-Chief). Handbook of Psychology.

[KLH*03] KLATZKY R. L., LEDERMAN S. J., HAMILTON C., GRINDLEY M., SWENDSEN R. H.: Feeling textures through a probe: Effects of probe and surface geometry and exploratory factors. *Perception and Psychophysics 65(4)* (2003), pp. 613–631.

[KLM02a] KIM Y. J., LIN M. C., MANOCHA D.: DEEP: an incremental algorithm for penetration depth computation between convex polytopes. *Proc. of IEEE Conference on Robotics and Automation* (2002), 921–926.

[KLM02b] KIM Y. J., LIN M. C., MANOCHA D.: Fast penetration depth computation using rasterization hardware and hierarchical refinement. *Proc. of Workshop on Algorithmic Foundations of Robotics* (2002).

[KLM04] KIM Y. J., LIN M., MANOCHA D.: Incremental penetration depth estimation between convex polytopes using dual-space expansion. *IEEE Trans. on Visualization and Computer Graphics 10*, 1 (2004), 152–164.

[KOLM03] KIM Y. J., OTADUY M. A., LIN M. C., MANOCHA D.: Six-degree-of-freedom haptic rendering using incremental and localized computations. *Presence 12*, 3 (2003), 277–295.

[KSE*97] KUSCHFELDT S., SCHULZ M., ERTL T., REUDING T., HOLZNER M.: The use of a virtual environment for FE analysis of vehicle crashworthiness. In *Virtual Reality Annual International Symposium VRAIS* (Albuquerque, USA, March 1997), p. 209.

[KT99] KALLMANN M., THALMANN D.: Direct 3D interaction with smart objects. In *VRST* (London, December 1999), ACM Press.

[KW03] KOMERSKA R., WARE C.: Haptic task constraints for 3D interaction. In *IEEE Symposium on Virtual Reality* (2003).

[KWF*01] KUNIMATSUA A., WATANABE Y., FUJII H., SAITO T., HIWADA K., TAKAHASHI T., UEKI H.: Rendering of natural phenomena fast simulation and rendering techniques for fluid objects. *Computer Graphics Forum 20*, 3 (2001), 57–67.

[Lab99] LABEIN: DATum geometric modeller PDriver user's manual. Version 15.0, 1999.

[Lar01] LARSEN E.: A robot soccer simulator: A case study for rigid body contact. *Game Developers Conference* (2001).

[LC91] LIN M. C., CANNY J. F.: Efficient algorithms for incremental distance computation. In *Proc. IEEE Internat. Conf. Robot. Autom.* (1991), vol. 2, pp. 1008–1014.

[LCN99] LOMBARDO J. C., CANI M.-P., NEYRET F.: Real-time collision detection for virtual surgery. *Proc. of Computer Animation* (1999).

[LDC05] LONGHURST P., DEBATTISTA K., CHALMERS A.: Snapshot: A rapid technique for selective global illumination rendering. In *Thirteenth International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision* (2005).

[LDW97] LOUNSBERY M., DEROSE T. D., WARREN J.: Multiresolution analysis for surfaces of arbitrary topological type. *ACM Trans. Graph. 16*, 1 (Jan. 1997), 34–73.

[LE97] LUEBKE D., ERIKSON C.: View-dependent simplification of arbitrary polygon environments. In *Proc. of ACM SIGGRAPH* (1997).

[Led74] LEDERMAN S. J.: Tactile roughness of grooved surfaces: The touching process and the effects of macro- and microsurface structure. *Perception and Psychophysics 16* (1974), pp. 385–395.

[Lem65] LEMKE C. E.: Bimatrix equilibrium points and mathematical programming. *Management Science 11* (1965), 681–689.

[LG98] LIN M., GOTTSCHALK S.: Collision detection between geometric models: A survey. *Proc. of IMA Conference on Mathematics of Surfaces* (1998).

[LGLM00] LARSEN E., GOTTSCHALK S., LIN M., MANOCHA D.: Distance queries with rectangular swept sphere volumes. *Proc. of IEEE Int. Conference on Robotics and Automation* (2000).

[LH01] LUBEKE D., HALLEN B.: Perceptually driven simplification for interactive rendering. In *Twelveth Eurographics Workshop on Rendering* (2001), pp. 221–223.

[Lin93] LIN M.: *Efficient Collision Detection for Animation and Robotics*. PhD thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley, December 1993.

[LKHG00] LEDERMAN S. J., KLATZKY R. L., HAMILTON C., GRINDLEY M.: Perceiving surface roughness through a probe: Effects of applied force and probe diameter. *Proceedings of the ASME DSCD-IMECE* (2000).

[LKHR99] LEDERMAN S. J., KLATZKY R. L., HAMILTON C., RAMSAY G. I.: Perceiving roughness via a rigid stylus: Psychophysical effects of exploration speed and mode of touch. *Haptics-e* (1999).

[LLW04] LIU Y., LIU X., WU E.: Real-time 3D fluid simulation on GPU with complex obstacles. In *Computer Graphics and Applications Twelveth Pacific Conference* (2004), pp. 247–256.

[LM99] LOSCHKY L. C., MCCONKIE G. W.: Gaze contingent displays: Maximizing display bandwidth efficiency. In *ARL Federated Laboratory Advanced Displays and Interactive Displays Consortium, Advanced Displays and Interactive Displays Third Annual Symposium* (1999), pp. 79–83.

[LM04] LIN M. C., MANOCHA D.: Collision and proximity queries. In *Handbook of Discrete and Computational Geometry, 2nd Ed.*, Goodman J. E., O'Rourke J., (Eds.). CRC Press LLC, Boca Raton, FL, 2004, ch. 35, pp. 787–807.

[LMYM01] LOSCHKY L. C., MCCONKIE G. W., YANG J., MILLER M. E.: Perceptual effects of a gaze-contingent multi-resolution display based on a model of visual sensitivity. In *ARL Federated Laboratory 5th Annual Symposium - ADID Consortium* (2001), pp. 53–58.

[Lot84] LOTSTEDT P.: Numerical simulation of time-dependent contact friction problems in rigid body mechanics. *SIAM Journal of Scientific Statistical Computing 5* (1984), 370–393.

[LRC*02] LUEBKE D., REDDY M., COHEN J., VARSHNEY A., WATSON B., HUEBNER R.: *Level of Detail for 3D Graphics*. Morgan-Kaufmann, 2002.

[LRW*01] LUBEKE D., REDDY M., WATSON B., COHEN J., VARSHNEY A.: Advanced issues in level of detail. In *SIGGRAPH: Course* (Los Angeles, August 2001), ACM Press.

[LS91] LAMOTTE R. H., SRINIVASAN M. A.: Surface microgeometry: Tactile perception and neural encoding. In *Information Processing in the Somatosensory System*, Franzen O., Westman J., (Eds.). Macmillan Press, London, 1991, pp. 49–58.

[LT98] LINDSTROM P., TURK G.: Fast and memory efficient polygonal simplification. *Proc. of IEEE Visualization* (1998), 279–286.

[LVK02] LARSSON P., VÄSTFJÄLL D., KLEINER M.: Better presence and performance in virtual environments by improved binaural sound rendering. In *AES Twenty Second International Conference on Virtual Synthetic and Entertainment Audio* (Espoo, Finland, June 2002), pp. 31–38.

[Man01a] MANNINEN T.: Rich interaction in the context of networked virtual environments - experiences gained from the multi-player games domain. In *Joint HCI and IHM Conference* (2001), Blanford A., Vanderdonckt J.,, Gray P., (Eds.), pp. 383–398.

[Man01b] MANNINEN T.: Virtual team interactions in networked multimedia games case: "Counter-Strike" – Multi-player 3D action game. In *Presence Conference* (Phiadelphia, May 2001).

[Mar02] MARSH J.: *A Software Architecture for Interactive Multiuser Visualisation*. PhD thesis, The University of Manchester, 2002.

[Mau00] MAUVE M.: Consistency in replicated continuous interactive media. In *ACM Conference on Computer Supported Cooperative Work (CSCW)* (Philadelphia, PA, December 2000), pp. 181–190.

[MBB*02] MONTGOMERY K., BRUYNS C., BROWN J., SORKIN S., MAZZELLA F., THONIER G., TELLIER A., LERMAN B., MENON A.: Spring: A general framework for collaborative, real-time surgical simulation. In *Medicine Meets Virtual Reality* (Amsterdam, 2002), IOS Press.

[MBCT98] MUSSE S. R., BABSKI C., CAPIN T., THALMANN D.: Crowd modelling in collaborative virtual environments. In *VRST* (Taipei, Taiwan, November 1998), pp. 115–124.

[MC95] MIRTICH B., CANNY J.: Impulse-based simulation of rigid bodies. In *Symposium on Interactive 3D Graphics* (1995), ACM Press.

[MCF99] MILLER B. E., COLGATE J. E., FREEMAN R. A.: Guaranteed stability of haptic systems with nonlinear virtual environments. *IEEE Transactions on Robotics and Automation 16*, 6 (1999), 712–719.

[MCTG00] MCNAMARA A., CHALMERS A., TROSCIANKO T., GILCHRIST I.: Comparing real and synthetic scenes using human judgements of lightness. In *Eurographics Workshop on Rendering* (Brno, June 2000).

[MD02] MARMITT G., DUCHOWSKI A. T.: Modeling visual attention in VR: Measuring the accuracy of predicted scanpaths. In *Eurographics: Short Presentations* (2002), pp. 217–226.

[MGFB01] MOTTET D., GUIARD Y., FERRAND T., BOOTSMA R. J.: Two-handed performance of a rhythmical fitts task by individuals and dyads. *Experimental Psychology: Human Perception and Performance 27*, 6 (2001), 1275–1286.

[MGP*04] MARSH J., GLENCROSS M., PETTIFER S., HUBBOLD R., COOK J., DAUBRENET S.: Minimising latency and maintaining consistency in distributed virtual prototyping. In *ACM SIGGRAPH Conference on the Virtual Reality Continuum and its Applications in Industry (VRCAI)* (Singapore, June 2004), pp. 386–389.

[MGPH05] MARSH J., GLENCROSS M., PETTIFFER S., HUBBOLD R.: A robust network architecture supporting rich behaviour in collaborative interactive applications. *In Review for IEEE TVCG Special Issue on Special Issue-Haptics, Virtual and Augmented Reality* (2005).

[MHS02] MCLAUGHLIN M., HESPANHA J. P., SUKHATME G. S.: *Touch in Virtual Environments*. Prentice Hall, 2002.

[Mic63] MICHOTTE A. E.: *The Perception of Causality*. Basic Books, 1963.

[Min95] MINSKY M.: *Computational Haptics: The Sandpaper System for Synthesizing Texture for a Force-Feedback Display*. PhD thesis, Ph.D. Dissertation, Program in Media Arts and Sciences, MIT, 1995. Thesis work done at UNC-CH Computer Science.

[Mir96] MIRTICH B. V.: *Impulse-based Dynamic Simulation of Rigid Body Systems*. PhD thesis, University of California, Berkeley, 1996.

[Mir98a] MIRTICH B.: V-Clip: Fast and robust polyhedral collision detection. *ACM Transactions on Graphics 17*, 3 (July 1998), 177–208.

[Mir98b] MIRTICH B. V.: *Rigid Body Contact: Collision Detection to Force Computation*. Tech. Rep. TR98-01, Mitsubishi Electric Research Laboratory, 1998.

[Mir00] MIRTICH B.: Timewarp rigid body simulation. In *ACM SIGGRAPH Computer Graphics* (New Orleans, LA, 2000), pp. 193–200.

[MIWJ02] MEEHAN M., INSKO B., WHITTON M., JR. F. P. B.: Physiological measures of presence in stressful virtual environments. In *ACM SIGGRAPH Transactions on Graphics* (2002), pp. 645–652.

[ML97] MCCONKIE G. W., LOSCHKY L. C.: Human performance with a gaze-linked multi-resolutional display. In *ARL Federated Laboratory Advanced Displays and Interactive Displays Consortium, Advanced Displays and Interactive Displays First Annual Symposium* (1997), pp. 25–34.

[MMF03] MARCELINO L., MURRAY N., FERNANDO T.: A constraint manager to support virtual maintainability. In *Computers and Graphics* (2003), vol. 27, pp. 19–26.

[MOS*90] MINSKY M., OUH-YOUNG M., STEELE O., BROOKS, JR. F. P., BEHENSKY M.: Feeling and seeing: Issues in force display. In *Computer Graphics (1990 Symposium on Interactive 3D Graphics)* (Mar. 1990), Riesenfeld R., Sequin C., (Eds.), vol. 24, pp. 235–243.

[MPPW94] MUSGRAVE F. K., PEACHEY D., PERLIN K., WORLEY S.: *Texturing and Modeling A Procedural Approach*. Academic Press Limited, 1994.

[MPT99] MCNEELY W., PUTERBAUGH K., TROY J.: Six degree-of-freedom haptic rendering using voxel sampling. *Proc. of ACM SIGGRAPH* (1999), 401–408.

[MPW99] MARSH J., PETTIFER S., WEST A.: A technique for maintaining continuity of experience in networked virtual environments. In *UKVRSIG* (Sept. 1999).

[MQW01] MCDONNELL K., QIN H., WLODARCZYK R.: Virtual clay: A real-time sculpting system with haptic interface. *Proc. of ACM Symposium on Interactive 3D Graphics* (2001), 179–190.

[MR98] MACK A., ROCK I.: *Inattentional Blindness*. MIT Press, 1998.

[MR04] MANIA K., ROBINSON A.: The effect of quality of rendering on user lighting impressions and presence in virtual environments. In *ACM SIGGRAPH International conference on Virtual Reality Continuum and its Applications in Industry (VRCAI)* (Singapore, 2004).

[MRC*86] MEYER G. W., RUSHMEIER H. E., COHEN M. F., GREENBERG D. P., TORRANCE K. E.: An experimental evaluation of computer graphics imagery. *ACM Transactions on Graphics 5*, 1 (1986), 30–50.

[MRF*96] MARK W., RANDOLPH S., FINCH M., VAN VERTH J., TAYLOR II R. M.: Adding force feedback to graphics systems: Issues and solutions. In *SIGGRAPH 96 Conference Proceedings* (1996), Rushmeier H., (Ed.), Annual Conference Series, pp. 447–452.

[MRWJ03] MEEHAN M., RAZZAQUE S., WHITTON, JR. F. P. B.: Effect of latency on presence in stressful virtual environments. In *IEEE Virtual Reality* (2003), pp. 141–148.

[MS94] MASSIE T. M., SALISBURY J. K.: The phantom haptic interface: A device for probing virtual objects. *Proc. of ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems 1* (1994), 295–301.

[MS95] MACIEL P. W. C., SHIRLEY P.: Visual navigation of large environments using textured clusters. In *Symposium on Interactive 3D Graphics* (1995), pp. 95–102.

[MS01] MILENKOVIC V. J., SCHMIDL H.: Optimization-based animation. *SIGGRAPH 01 Conference Proceedings* (2001), 37–46.

[MTAS01] MYSZKOWSKI K., TAWARA T., AKAMINE H., SEIDEL H.-P.: Perception-guided global illumination solution for animation rendering. In *SIGGRAPH* (2001), ACM Press, pp. 221–230.

[MTHC03] MANIA K., TROSCIANKO T., HAWKES R., CHALMERS A.: Fidelity metrics for virtual environment simulations based on spatial memory awareness states. *Presence: Teleoperators and Virtual Environments 12*, 3 (2003), 296–310.

[Mus89] MUSGRAVE F.: Prisms and rainbows: A dispersion model for computer graphics. In *Proceedings of the Graphics Interface Vision Interface* (Toronto, Ontario, June 1989).

[MVHE04] MAUVE M., VOGEL J., HILT V., EFFELSBERG W.: Local-lag and timewarp: Providing consistency for replicated continuous applications. *IEEE Transactions on Multimedia 6*, 1 (February 2004), 47–57.

[MW88] MOORE M., WILHELMS J.: Collision detection and response for computer animation. In *Computer Graphics (SIGGRAPH '88 Proceedings)* (Aug. 1988), Dill J., (Ed.), vol. 22, pp. 289–298.

[MW93] MACKENZIE I. S., WARE C.: Lag as a determinant of human performance in interactive systems. In *INTERCHI Proceedings of the ACM Conference on Human Factors in Computing Systems* (New York, 1993), pp. 488–493.

[MYDN01] MIYAZAKI R., YOSHIDA S., DOBASHI Y., NISHITA T.: A method for modelling clouds cased on atmospheric fluid dynamics. In *Pacific Graphics Conference* (2001), IEEE Computer Society, pp. 363–372.

[Mys98] MYSZKOWSKI K.: The visible differences predictor: Applications to global illumination problems. In *Eurographics Workshop on Rendering* (1998), pp. 223–226.

[MZP*94] MACEDONIA M. R., ZYDA M. J., PRATT D. R., BARHAM P. T., ZESWITZ S.: NPSNET: A network software architecture for large-scale ves. *Presence Teleoperators and Virtual Environments 3*, 4 (1994), 265–287.

[NJC99] NELSON D. D., JOHNSON D. E., COHEN E.: Haptic rendering of surface-to-surface sculpted model interaction. *Proc. of ASME Dynamic Systems and Control Division* (1999).

[OC99] OKAMURA A., CUTKOSKY M.: Haptic exploration of fine surface features. *Proc. of IEEE Int. Conf. on Robotics and Automation* (1999), pp. 2930–2936.

[OC01] OKAMURA A. M., CUTKOSKY M. R.: Feature detection for haptic exploration with robotic fingers. *International Journal of Robotics Research 20*, 12 (2001), 925–938.

[OD01] O'SULLIVAN C., DINGLIANA J.: Collisions and perception. *ACM Trans. on Graphics 20*, 3 (2001), pp. 151–168.

[ODGK03] O'SULLIVAN C., DINGLIANA J., GIANG T., KAISER M. K.: Evaluating the visual fidelity of physically based animations. *Proc. of ACM SIGGRAPH* (2003), 527–536.

[OJSL04] OTADUY M. A., JAIN N., SUD A., LIN M. C.: Haptic display of interaction between textured models. *Proc. of IEEE Visualization* (2004), 297–304.

[OL03a] OTADUY M. A., LIN M. C.: CLODs: Dual hierarchies for multiresolution collision detection. *Eurographics Symposium on Geometry Processing* (2003), 94–101.

[OL03b] OTADUY M. A., LIN M. C.: Sensation preserving simplification for haptic rendering. In *ACM SIGGRAPH Transactions on Graphics* (San Diego, CA, 2003), vol. 22, pp. 543–553.

[OL04] OTADUY M. A., LIN M. C.: A perceptually-inspired force model for haptic texture rendering. *Proc. of Symposium APGV* (2004), 123–126.

[OL05] OTADUY M. A., LIN M. C.: Stable and responsive six-degree-of-freedom haptic manipulation using implicit integration. *Proc. of World Haptics Conference* (2005), 247–256.

[OMJ*03] OLIVEIRA M., MORTENSEN J., JORDAN J., STEED A., SLATER M.: Considerations in the design of virtual environment systems: A case study. In *Second International Conference on Application and Development of Computer Games* (Hong Kong, January 2003).

[ORC99] O'SULLIVAN C., RADACH R., COLLINS S.: A model of collision perception for real-time animation. *Computer Animation and Simulation* (1999), pp. 67–76.

[OY90] OUH-YOUNG M.: *Force Display in Molecular Docking*. PhD thesis, University of North Carolina, Computer Science Department, 1990.

[PC99a] PESHKIN M., COLGATE J. E.: Cobots. *Industrial Robot 26*, 5 (1999), 335–341.

[PC99b] POPE J., CHALMERS A.: Multi-sensory rendering: Combining graphics and acoustics. In *Seventh International Conference in Central Europe on Computer Graphics* (1999).

[PCMT01] PETTIFER S., COOK J., MARIANI J., TREVOR J.: Exploring realtime visualisation of large abstract data spaces with QSPACE. In *IEEE Virtual Reality* (Yokohama, Japan, March 2001), pp. 293–294.

[PCMW00] PETTIFER S., COOK J., MARSH J., WEST A.: DEVA3: Architecture for a large-scale distributed virtual reality system. In *ACM Symposium on Virtual Reality Software and Technology* (Seoul, Korea, 2000), pp. 33–40.

[PDJ*01] PAI D. K., DOEL K. V., JAMES D. L., LANG J., LLOYD J. E., RICHMOND J. L., YAU S. H.: Scanning physical interaction behavior of 3d objects. *Computer Graphics (Proc. of ACM SIGGRAPH)* (2001).

[Pet99] PETTIFER S. R.: *An operating environment for large scale virtual reality*. PhD thesis, The University of Manchester, 1999.

[PFG98] PATTANAIK S. N., FAIRCHILD J. F. M. D., GREENBERG D. P.: A multiscale model of adaptation and spatial vision for realistic image display. In *SIGGRAPH* (1998), ACM Press, pp. 287–298.

[PJC04] POTTER K., JOHNSON D., COHEN E.: Height field haptics. *Proc. of Haptics Symposium* (2004).

[PK99] PARK K. S., KENYON R. V.: Effects of network characteristics on human performance in a collaborative virtual environment. In *IEEE Virtual Reality* (1999), IEEE Computer Society, pp. 104–111.

[PM01] PETTIFER S., MARSH J.: A collaborative access model for shared virtual environments. In *Tenth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises WETICE* (2001), pp. 257–262.

[PMS*99] PARKER S., MARTIN W., SLOAN P. P., SHIRLEY P., SMITS B., HANSEN C.: Interactive ray tracing. In *Symposium on Interactive 3D Computer Graphics* (April 1999).

[PR97] PAI D. K., REISSEL L. M.: Haptic interaction with multiresolution image curves. *Computer and Graphics 21* (1997), 405–411.

[Qui94] QUINLAN S.: Efficient distance computation between non-convex objects. In *Proceedings of International Conference on Robotics and Automation* (1994), pp. 3324–3329.

[RAB*00] ROSE F. D., ATTREE E. A., BROOKS B. M., PARSLOW D. M., PENN P. R.: Training in virtual environments: Transfer to real world tasks and equivalence to real task training. *Ergonomics 43*, 4 (April 2000), 494–511.

[Red97] REDDY M.: *Perceptually Modulated Level of Detail for Virtual Environments (CST- 134-97)*. PhD thesis, University of Edinburgh, 1997.

[Rei04] REINER M.: The role of haptics in immersive telecommunication environments. *IEEE Transactions on Circuits and Systems for Video Technology 14*, 3 (March 2004), 392–401.

[Rey87] REYNOLDS C. W.: Flocks, herds, and schools: A distributed behavioral model. In *SIGGRAPH Computer Graphics* (1987), vol. 21, pp. 25–34.

[RG98] ROESSLER A., GRANTZ V.: Performance evaluation of input devices in virtual environments. In *IFIP Working Group Conference on Designing Effective and Usable Multimedia Systems* (1998), pp. 68–78.

[RGA95] ROLLAND J. P., GIBSON W., ARIELY D.: Towards quantifying depth and size perception in virtual environments. *Presence: Teleoperators and Virtual Environments 4*, 1 (1995), 24–49.

[RJ99] RUHLEDER K., JORDAN B.: Meaning-making across remote sites: How delays in transmission affect interaction. In *Sixth European Conference on Computer-Supported Cooperative Work ECSCW* (Copenhagen, Denmark, 1999), Kulwer, pp. 411–429.

[RK63] RIESZ R., KLEMMER E. T.: Subjective evaluation of delay and echo supressors in telephone communications. *Bell System Technical Journal 42* (November 1963), 2919–2933.

[RK00] RUSPINI D., KHATIB O.: 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] REDON S., KHEDDAR A., COQUILLART S.: Fast continuous collision detection between rigid bodies. *Proc. of Eurographics (Computer Graphics Forum)* (2002).

[RKK97] RUSPINI D., KOLAROV K., KHATIB O.: The haptic display of complex graphical environments. *Proc. of ACM SIGGRAPH* (1997), 345–352.

[RLP*95] RUSHMEIER H., LARSON G., PIATKO C., SANDERS P., RUST B.: Comparing real and synthetic images: Some ideas about metrics. In *Eurographics Rendering Workshop* (June 1995), pp. 82–91.

[RPG99] RAMASUBRAMANIAN M., PATTANAIK S. N., GREENBERG D. P.: A perceptually based physical error metric for realistic image synthesis. In *SIGGRAPH* (1999), ACM Press, pp. 73–82.

[RPP*01] RENZ M., PREUSCHE C., PÖTKE M., KRIEGEL H.-P., HIRZINGER G.: Stable haptic interaction with virtual environments using an adapted voxmap-pointshell algorithm. *Eurohaptics Conference* (2001).

[RV64] ROCK I., VICTOR J.: Vision and touch: An experimentally created conflict between the two senses. *Science 143* (1964), pp. 594–596.

[Sal99] SALISBURY J. K.: Making graphics physically tangible. *Communications of the ACM 42*, 8 (1999).

[SBM*95] SALISBURY K., BROCK D., MASSIE T., SWARUP N., ZILLES C.: Haptic rendering: Programming touch interaction with virtual objects. In *1995 Symposium on Interactive 3D Graphics* (Apr. 1995), Hanrahan P., Winget J., (Eds.), ACM SIGGRAPH, pp. 123–130. ISBN 0-89791-736-7.

[SBNG03] SHEN X., BOGSANYI F., NI L., GEORGANAS N. D.: A heterogeneous scalabale architecture for collaborative haptics environments. In *IEEE International Workshop on Haptic Audio and Visual Environments (HAVE)* (September 2003), pp. 113–118.

[SC99] SIMONS D., CHABRIS C.: Gorillas in our midst: Sustained inattentional blindness for dynamic events. *Perception 28* (1999), 1059–1074.

[SCCD04] SUNDSTEDT V., CHALMERS A., CATER K., DEBATTISTA K.: Top-down visual attention for efficient rendering of task related scenes. In *Vision, Modeling and Visualization* (2004).

[SDL*05] SUNDSTEDT V., DEBATTISTA K., LONGHURST P., CHALMERS A., TROSCIANKO T.: Visual attention for efficient high-fidelity graphics. In *Spring Conference on Computer Graphics SCCG* (May 2005).

[SDS96] STOLLNITZ E., DEROSE T., SALESIN D.: *Wavelets for Computer Graphics: Theory and Applications*. Morgan-Kaufmann, 1996.

[Sei90] SEIDEL R.: Linear programming and convex hulls made easy. In *Proc. 6th Ann. ACM Conf. on Computational Geometry* (Berkeley, California, 1990), pp. 211–215.

[SF93] STAM J., FIUME E.: Turbulent wind fields for gaseous phenomena. In *Computer Geraphics* (August 1993), vol. 9, pp. 369–376.

[SG01] SHIRMOHAMMADI S., GEORGANAS N. D.: An end-to-end communication architecture for collaborative virtual environments. *Computer Networks 35*, 2-3 (2001), 351–367.

[SGG*00] SANDER P. V., GU X., GORTLER S. J., HOPPE H., SNYDER J.: Silhouette clipping. *Proc. of ACM SIGGRAPH* (2000), 327–334.

[Sha98] SHAW J.: Distributed legible city. In *IST 98 — Information Society Technologies Conference and Exhibition* (Vienna, Austria, 1998).

[Shi92] SHIMOGA K.: Finger force and touch feedback issues in dextrous manipulation. *NASA-CIRSSE International Conference on Inetelligent Robotic Systems for Space Exploration* (1992).

[Sla99] SLATER M.: Measuring presence: A response to the witmer and singer presence questionnaire. *Presence Teleoperators and Virtual Environments 8*, 5 (1999), 560–565.

[SP94] SILLION F. X., PUECH C.: *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco, CA, USA, 1994.

[SP96] SIIRA J., PAI D. K.: Haptic textures – a stochastic approach. *Proc. of IEEE International Conference on Robotics and Automation* (1996), 557–562.

[SPD00] SPENCE C., PAVANI F., DRIVER J.: Crossmodal links between vision and touch in covert endogenous spatial attention. *Journal of Experimental Psychology: Human Perception and Performance 26* (2000), pp. 1298–1319.

[SRGS00] SALLNÄS E.-L., RASSMUS-GROEHN K., SJOESTRÖM C.: Supporting presence in collaborative environments by haptic force feedback. *ACM Transactions on Computer-Human Interaction 7*, 4 (2000), 461–476.

[SSGH01] SANDER P. V., SNYDER J., GORTLER S. J., HOPPE H.: Texture mapping progressive meshes. *Proc. of ACM SIGGRAPH* (2001), 409–416.

[SSM*99] STEVENSON D. R., SMITH K. A., MCLAUGHLIN J. P., GUNN C. J., VELDKAMP J. P., DIXON M. J.: Haptic workbench: A multisensory virtual environment. In *SPIE* (1999), vol. 3639, pp. 356–366.

[SSU00] SLATER M., SADAGIC A., USOH M.: Small group behaviour in a virtual and real time environment: A comparative study. *Presence: Teleoperators and Virtual Environments 9*, 1 (2000), 37–51.

[ST96] STEWART D. E., TRINKLE J. C.: An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal of Numerical Methods in Engineering 39* (1996), 2673–2691.

[ST00] STEWART D. E., TRINKLE J. C.: An implicit time-stepping scheme for rigid body dynamics with coulomb friction. *IEEE International Conference on Robotics and Automation* (2000), 162–169.

[Sta00] STAM J.: Interacting with smoke and fire in real-time. *Communications of the ACM 43*, 7 (July 2000), 76–83.

[SU93] SLATER M., USOH M.: *An Experimental Exploration of Presence in Virtual Environments*. Tech. Rep. 689, Department of Computer Science, University College London, 1993.

[Sut65] SUTHERLAND I.: The ultimate display. *Proc. of IFIP* (1965), 506–508.

[SV94] SALCUDEAN S. E., VLAAR T. D.: 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).

[SW97] SLATER M., WILBUR S.: A framework for immersive virtual environments (FIVE): speculations on the role of presence in virtual environments. *Presence: Teleoperators and Virtual Environments 6*, 6 (1997), 603–616.

[SZ99] SINGHAL S., ZYDA M.: *Networked Virtual Environments Design and Implementation*. ACM Press SIGGRAPH Series Addison Wesley, 1999.

[SZ03] SALLÄS E.-L., ZHAI S.: Collaboration meets fitts' law: Passing virtual objects with and without haptic force feedback. In *INTERACT IFIP Conference on Human-Computer Interaction* (2003), pp. 97–104.

[Tau95] TAUBIN G.: A signal processing approach to fair surface design. In *SIGGRAPH 95 Conference Proceedings* (Aug. 1995), Cook R., (Ed.), Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 351–358. held in Los Angeles, California, 06-11 August 1995.

[TF88] TERZOPOULOS D., FLEISCHER K.: Modelling in-elastic deformation: Viscoelasticity, plasticity, fracture. In *SIGGRAPH* (1988), ACM Press, pp. 269–278.

[TF98] TERZOPOULOS D., FLEISCHER K.: Deformable models. *Th Visual Computer 4* (1998), 306–331.

[TGD04] TSINGOS N., GALLO E., DRETTAKIS G.: Perceptual audio rendering of complex virtual environments. In *ACM SIGGRAPH Transactions on Graphics* (August 2004), vol. 23 (3), pp. 249–258.

[TJC97] THOMPSON T. V., JOHNSON D., COHEN E.: Direct haptic rendering of sculptured models. *Proc. of ACM Symposium on Interactive 3D Graphics* (1997), pp. 167–176.

[TRC*93] TAYLOR R. M., ROBINETT W., CHII V., BROOKS F., WRIGHT W.: The nanomanipulator: A virtual-reality interface for a scanning tunneling microscope. In *Proc. of ACM SIGGRAPH* (1993), pp. 127–134.

[TSRD01] TORKINGTON J., SMITH S., REES B., DARZI A.: Skill transfer from virtual reality to a real laparoscopic task. *Surgical Endoscopy 15*, 10 (May 2001), 1076–1079.

[UNB*02] UNGER B. J., NICOLAIDIS A., BERKELMAN P. J., THOMPSON A., LEDERMAN S. J., KLATZKY R. L., HOLLIS R. L.: Virtual peg-in-hole performance using a 6-dof magnetic levitation haptic device: Comparison with real forces and with visual guidance alone. *Proc. of Haptics Symposium* (2002), 263–270.

[UT01] ULICNY B., THALMANN D.: Crowd simulation for interactive virtual environments and VR training systems. In *Eurographics workshop on Animation and Simulation* (2001), Springer-Verlag, pp. 163–170.

[UT02] ULICNY B., THALMANN D.: Towards interactive real-time crowd behavior simulation. *Computer Graphics Forum 21*, 4 (December 2002), 767–775.

[VMKK00] VOLEVICH V., MYSZKOWSKI K., KHODULEV A., KOPYLOV E. A.: Using the visual differences predictor to improve performance of progressive global illumination computations. *ACM Transactions on Graphics 19*, 2 (April 2000), 122–161.

[VRS05] Welding simulator. Online Document, May 2005. http://www.vrsim.net/Products/weldingsimmain.html.

[Wal05] WALLACE M.: Vrsim welding application. Personnal correspondance with CEO of VRSim, 2005.

[WDGT01] WU X., DOWNES M. S., GOTEKIN T., TENDICK F.: Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. In *Eurographics* (2001), Chalmers A., Rhyne T. M., (Eds.), vol. 20, pp. 349–358.

[Wei02a] WEISS H.: Dynamics of geometrically nonlinear rods: I. mechanical models and equations of motion. *Nonlinear Dynamics 30* (2002), 357–381.

[Wei02b] WEISS H.: Dynamics of geometrically nonlinear rods: II. mechanical models and equations of motion. *Nonlinear Dynamics 30* (2002), 383–415.

[Whi94] WHITEHOUSE D. J.: *Handbook of Surface Metrology*. Institute of Physics Publishing, Bristol, 1994.

[Win81] WINTER A. B. P. R.: Theory and application of finite element analysis to structural crash simulation. *Computers and Structures 13* (June 1981), 277–285.

[WKB*02] WALD I., KOLLIG T., BENTHIN C., KELLER A., SLUSALLEK P.: Interactive global illumination using fast ray tracing. In *Thirteenth Eurographics workshop on Rendering* (Pisa, Italy, 2002), Eurographics Association, pp. 15–24. ray tracing, realistic rendering, real-time, global illumination.

[WKSKH99] WILSON J. P., KLINE-SCHODER R. J., KENTON M. A., HOGAN N.: Algorithms for network-based force feedback. In *The Fourth PHANTOM Users Group Workshop* (1999).

[WM03] WAN M., MCNEELY W. A.: Quasi-static approximation for 6 degrees-of-freedom haptic rendering. *Proc. of IEEE Visualization* (2003), 257–262.

[WS98] WITMER B. G., SINGER M. J.: Measuring presence in virtual environments: A presence questionnaire. *Presence Teleoperators and Virtual Environments 7*, 3 (1998), 225–240.

[WS03] WALKER S. P., SALISBURY J. K.: Large haptic topographic maps: Marsview and the proxy graph algorithm. *Proc. of ACM Symposium on Interactive 3D Graphics* (2003), pp. 83–92.

[WSBW01] WALD I., SLUSALLEK P., BENTHIN C., WAGNER M.: Interactive rendering with coherent ray tracing. *Computer Graphics Forum 20*, 3 (2001).

[WTR*03] WANG D., TUER K., ROSSI M., NI L., SHU J.: The effect of time delays on tele-haptics. In *Second IEEE Internatioal Workshop on Haptic, Audio and Visual Environments and Their Applications HAVE* (2003), pp. 7–12.

[Wu00] WU D.: Penalty methods for contact resolution. *Game Developers Conference* (2000).

[WZ98] WATSEN K., ZYDA M.: Bamboo – A portable system for dynamically extensible, real-time, networked, virtual environments. In *Virtual Reality Annual International Symposium* (1998), pp. 252–259.

[Yan96] YANTIS S.: Attentional capture in vision. In *Converging Operations in the Study of Selective Visual Attention*, Kramer A., Coles M.,, Logan G., (Eds.). American Psychological Association, 1996, pp. 45–76.

[Yar67] YARBUS A. L.: Eye movements during perception of complex objects. In *Eye Movements and Vision* (New York, 1967), Riggs L. A., (Ed.), Plenum Press, pp. 171–196.

[Yee00] YEE H.: *Spatiotemporal Sensitivity and Visual Attention for Efficient Rendering of Dynamic Environments*. Master's thesis, Cornell University, 2000.

[YPG01] YEE H., PATTANAIK S., GREENBERG D. P.: Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments. *ACM Transactions on Computer Graphics 20,* 1 (2001), 39–65.

[YSLM04] YOON S., SALOMON B., LIN M. C., MANOCHA D.: Fast collision detection between massive models using dynamic simplification. *Eurographics Symposium on Geometry Processing* (2004).

[ZF03] ZHANG Y., FERNANDO T.: 3D sound feedback act as task aid in a virtual assembly environment. In *Theory and Practice of Computer Graphics* (2003), pp. 209–214.

[ZS95] ZILLES C., SALISBURY K.: A constraint-based god object method for haptics display. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robotics and Systems* (1995).

[ZSS97] ZORIN D., SCHRÖDER P., SWELDENS W.: Interactive multiresolution mesh editing. *Proc. of ACM SIGGRAPH* (1997), 259–268.