Exploiting Perception in High-Fidelity Virtual Environments

Mashhuda Glencross, Alan G. Chalmers, Ming C. Lin, Miguel A. Otaduy and Diego Gutierrez

Contents

Abstract

The objective of this course is to provide an introduction to the issues that must be considered when building high-fidelity 3D engaging shared virtual environments. The principles of human perception guide important development of algorithms and techniques in collaboration, graphical, auditory, and haptic rendering. We aim to show how human perception is exploited to achieve realism in high fidelity environments within the constraints of available finite computational resources.

In this course we address the challenges faced when building such high-fidelity engaging shared virtual environments, especially those that facilitate collaboration and intuitive interaction. We present real applications in which such high-fidelity is essential. With reference to these, we illustrate the significant need for the combination of high-fidelity graphics in real time, better modes of interaction, and appropriate collaboration strategies.

After introducing the concept of high-fidelity virtual environments and why these convey important information to the user, we cover the main issues in two parts linked by the common thread of exploiting human perception. First we explore perceptually driven techniques that can be employed to achieve high-fidelity graphical rendering in real-time, and how incorporating authentic lighting effects helps to convey a sense of realism and scale in virtual re-constructions of historical sites.

Secondly, we examine how intuitive interaction between participants, and with objects in the environment, also plays a key role in the overall experience. How perceptual methods can be used to guide interest management and distribution choices, is considered with an emphasis on avoiding potential pitfalls when distributing physically-based simulations. An analysis of real network conditions and the implications of these for distribution strategies that facilitate collaboration is presented. Furthermore, we describe technologies necessary to provide intuitive interaction in virtual environments, paying particular attention to engaging multiple sensory modalities, primarily through physically-based sound simulation and perceptually high-fidelity haptic interaction.

The combination of realism and intuitive compelling interaction can lead to engaging virtual environments capable of exhibiting skills transfer, an illusive goal of many virtual environment applications.

Keywords: high-fidelity rendering, collaborative environments, virtual reality, multi-user, networked applications, human-computer interaction, perception, haptics

Part I

High-Fidelity Virtual Environments

1 Introduction

Virtual reality is a field of computer graphics research that has stemmed from Ivan Sutherland's vision of an "Ultimate Display" outlined in his seminal lecture in 1965 [Sut65]. He suggested a screen should be considered as a window upon a virtual world, and the challenge for computer graphics researchers is to make that world look, sound, respond to user interaction in real time, and even feel real. This implies active participation in a virtual environment which users perceive to be real.

Fred Brooks in his paper entitled "What's Real About Virtual Reality?" defined "a virtual reality experience as any in which the user is effectively immersed in a responsive virtual world. This implies user dynamic control of viewpoint" [Bro99]. Furthermore, he argued that technology had developed to the point at which Virtual Reality had become a realistic possibility.

For immersive environments, Brooks' view that component technologies have made big strides [Bro99] is fair. However, as both software and hardware improves, our expectations of the technology also becomes more demanding. Real-time performance requirements of virtual environments has resulted in the rendering requirements generally dominating computational costs – increasingly we want to see more realistic graphics. Thus many virtual environment demonstrator applications constitute walk-throughs involving limited direct interaction with the environment. The idea of a passive view upon a virtual environment is not strictly in accordance with Sutherlands's vision of virtual worlds in which users play an active role. The real world 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.

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 difficult 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? We suggest 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 behavioral fidelity. In the following sections we describe what we mean by these.

Figure 1: The UNC virtual pit experiment – Image courtesy of Fred Brooks, University of North Carolina Chapel Hill [MRWB03]

2 Compelling Virtual Environments

A particularly compelling example of a virtual environment, shown in Figure 1, is the University of North Carolina's 'Pit' model [MIWB02, MRWB03]. 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 behavioral fidelity of such environments becomes increasingly ambitious. Consequently, much recent research has focused on utilizing additional sensory stimuli to create compelling environments [DWS*99, SSM*99, HMG03, TGD04]. Auditory cues have been shown to increase participants' sense of presence [PC99b, LVK02] and studies have shown that in environments incorporating auditory feedback, subjects perform better in spatial localization tasks than in the absence of this stimulus [ZF03, TGD04].

Haptic feedback (conveying the sense of touch) has emerged as another important stimulus to exploit in many VR applications, and recently has been incorporated in a number of single user and shared demonstrator applications [MBB*02, GHAH03]. Many of these are simple testbed tasks used to assess the value of collaborative haptic manipulation [BOH97, Hub02, OMJ*03, Rei04, KKT*04].

2.1 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 also potentially has a causal relationship to the state of entities which the user influences in the environment. This state change may be reported back to the user through multi-sensory feedback such as color 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.

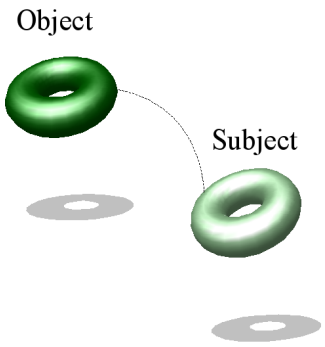


Figure 2: The Deva object/subject model

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 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 behaviorally-rich environments, containing a variety of dynamic entities whose state and/or structure may be changed. It is therefore useful to consider how rich behavior may be represented in frameworks for building virtual environments.

2.2 Behaviorally-Rich Shared Virtual Environments

Behavior-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 simulations need to be simplified [Gle00]. Provided such simplifications can still be used to compute plausible or perceptually correct behavior, 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 behaviors [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 behavior 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 behavioral 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 behavior in itself is insufficient. 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 behavioral 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 behaviorally-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 behavior 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 render and interact with the entities each participant's *client* creates corresponding subjective parts. These possess behavior and act plausibly until otherwise instructed by the server, consequently their state may subtly and briefly differ from the objective reality as shown in Figure 2. It is the responsibility of the objective component to update at appropriate intervals all its corresponding subjects, and this information may be customized on a per-entity basis [Pet99, Mar02].

Figures 3 to 6 and Figure 8 illustrate a number of example applications, implemented during large European funded projects, using the Deva system. The Placeworld application (Figures 3 and 4) was a world containing worlds, each individual world being 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 behavior 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 favorite exhibits, and the use of personalized avatars [Mar02].

The QPit application [PCMT01] was implemented to visualize 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 visualization of the semantic information within the graph data.

The Senet Game (shown in Figure 6) 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

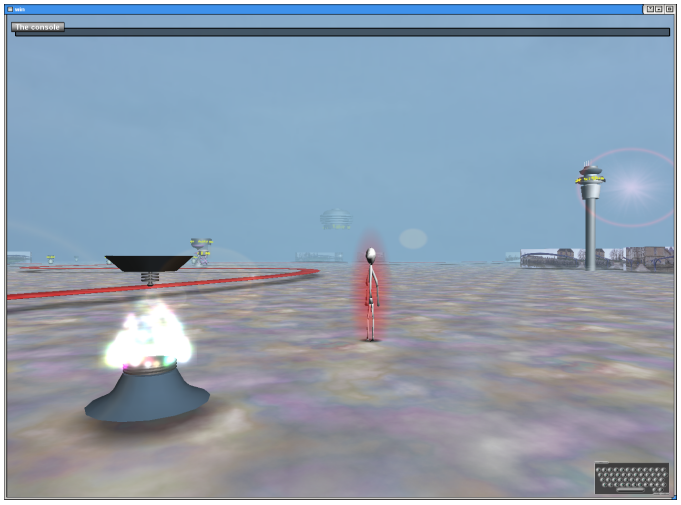


Figure 3: Placeworld – Image courtesy of James Marsh and Steve Pettifer, The University of Manchester [PM01]

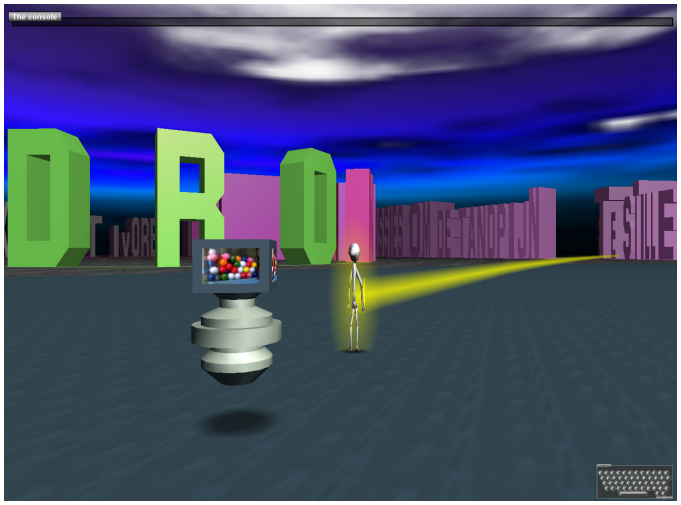


Figure 4: Placeworld legible city – Image courtesy of James Marsh and Steve Pettifer, The University of Manchester [Sha98, CP01]

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 behavior 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 behavior supported as core functionality, it is possible to rapidly develop and customize the behavior for a range of diverse applications. Physically based simulation techniques provide the most reliable method for computing realistic behavior 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].

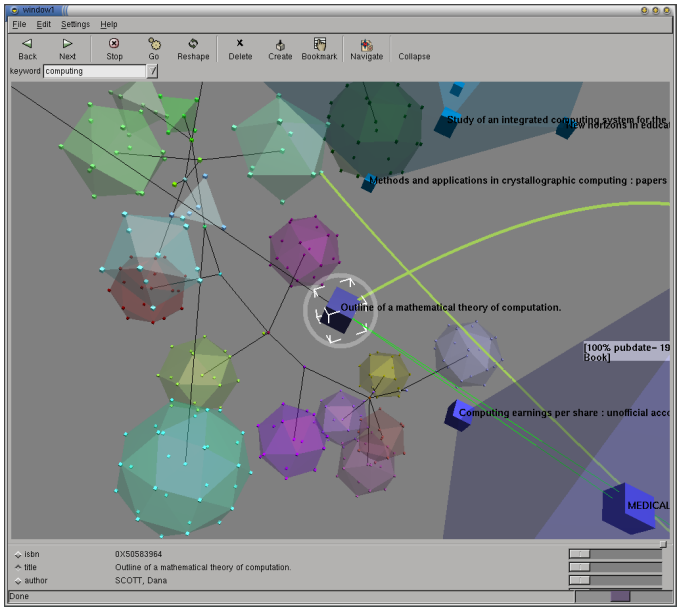


Figure 5: QPit – Image courtesy of James Marsh and Steve Pettifer, The University of Manchester [PCMT01]

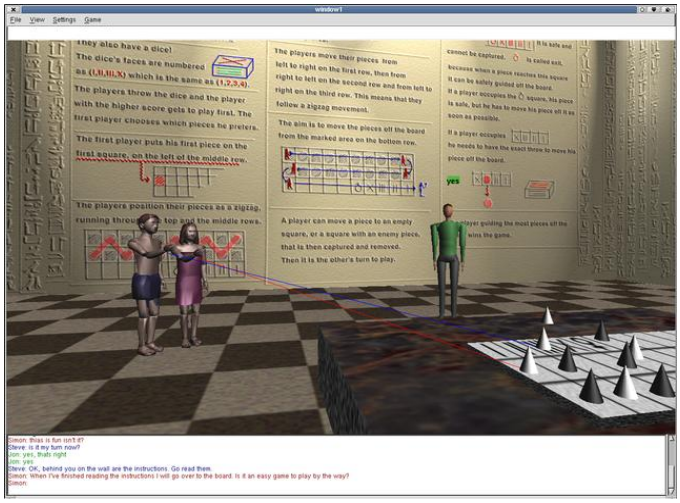


Figure 6: Senet game – Image courtesy of James Marsh and Steve Pettifer, The University of Manchester [EPMW99]

2.3 Advantages

Rich behavior 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 7). 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 similar to real welding equipment. Users wear a custom head-mounted display which is fitted into a welders mask, graphical, auditory and 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 Requirements

Brooks listed the following four technologies [Bro99] that are crucial for virtual reality:

- The immersive visual, auditory and haptic display

Figure 7: Virtual welding simulator – Image courtesy of FCS Control Systems, The Netherlands

- The graphics rendering system
- The tracking system
- The database to construct and maintain a model of the virtual world

We broaden these requirements to make them more suitable for both compelling and engaging desktop and immersive virtual reality. We argue that in view of developments in both software techniques and hardware capabilities that a combination of the following components are necessary for virtual reality:

- High-Fidelity graphics
- Complex interaction preferably engaging multiple sensory modalities
- Realistic simulation
- Tracking

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 visualizations of architectural designs. Graphical and auditory feedback may be sufficient for walk-throughs or simple visualizations, while haptic feedback may be required to assist in assembly tasks. Simple flocking behavior [Rey87] may suffice for describing the behavior of birds in synthetic landscapes, while more complex finite element models may be required to determine the deformation characteristics of materials in crash simulations [PW81, KSE*97, BFL*01]. Applications may also demand varying levels of tracking ranging from none to determining the position of the hand, head or full body.

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

3.1 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.

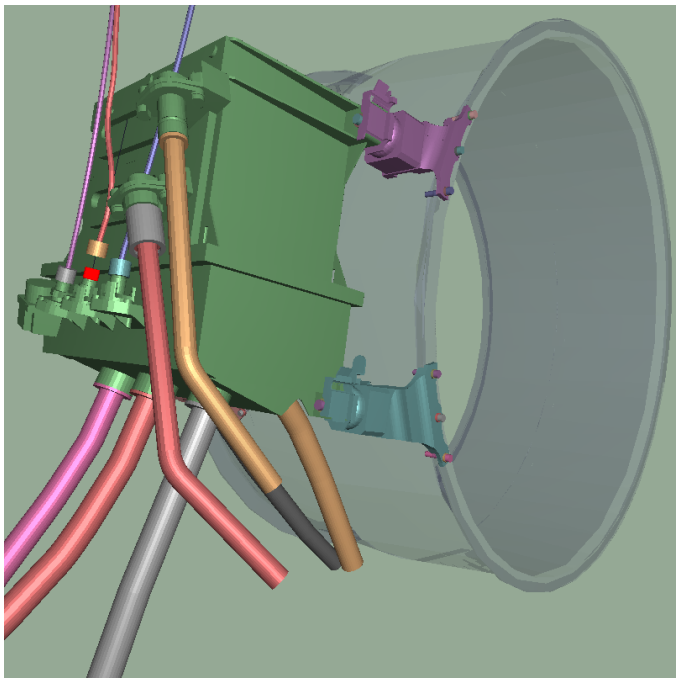


Figure 8: Collaborative CAD prototyping application [MGP*04]

3.2 High-Fidelity-Interaction

In these notes we consider high-fidelity interaction to include complex interaction engaging multiple sensory modalities and that occurring between multiple participants in a virtual environment.

3.2.1 Multiple Interaction Modalities

The most common modality used to provide feedback in virtual environments is the visual one, color 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 8 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.

Auditory cues were used in this application to convey information relating to collisions, geometric constraints, task and ownership. A range of different tunes were played to indicate if a particular constraint had been matched, was active, was met or broken. In order to mediate collaboration, subtle changes in background color were used to inform participants of transfer of control between them. 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, auditory feedback was also used to convey information relating to allowable interactions, for example when it was acceptable for other users to partake in a sequence.

The addition of haptic feedback offers a 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 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 VRSim's 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). Currently it is not practical to use this in virtual environments, however recent work in human computer interaction is beginning to exploit this modality [Kay04]. For certain kinds of training applications such as VRSim's welding simulator, it could be a useful modality to exploit as it could be used to convey additional information relating to the state of the material being used in the weld.

3.2.2 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. Bringing multiple participants into virtual environments contributes to the realism and level of engagement an application can offer. Environments populated by synthetic characters can become predictable, as users quickly recognize any preprogrammed sequences. Real participants customize 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 can be essential to completing the task. However when multiple participants are engaged in shared tasks, 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 these course notes, this can be non-trivial.

3.3 Complex Simulation

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

- Emergent behavior
- Material properties
- Natural phenomena

Emergent behavior involves a number of mainly independent entities each conforming to some simple particle physics, scripted responses, or ad-hoc goals. Combined, the behavior of these entities leads to a complex overall response. This type of simulation is used to support flocking behavior [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 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 behavior [BHW94, BW98, BWK03], and in applications such as crash [PW81, 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 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 [BTL90, Mus90, 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.

3.4 Tracking

If the only input devices employed to allow the user to interact with a virtual environment are a standard mouse (and/or keyboard) then mapping positional values for navigation purposes is straight-forward, and requires little more than obtaining 2D co-ordinates directly and mapping these to a position in 3D space. However, a range of more intuitive input devices can be employed in virtual environments that track the location in 3D of the user's hand, head or full-body in the real world, and then use this information to map users' movements to the virtual space. Tracking devices are generally based on magnetic, or optical technologies. The former suffer from magnetic field distortion caused by metal building materials, while the latter optical devices offer better accuracy. Ongoing research in the area of camera based tracking techniques offer the potential for un-intrusive tracking technology. Some tracking devices used in virtual environments are briefly discussed in §27.1.

4 Perception in Virtual Environments

The way in which a user perceives a virtual environment is closely linked to how compelling the environment may seem. How present users feel in virtual environments is difficult to measure and even the accepted measures are subjective [SW97, WS98, Sla99, SSU00] with physiological measures [MIWB02] perhaps offering more insight. The sense of presence can easily be lost and there may be any number of causes such as: the graphics do not appear to be correct or are not updated fast enough, a particular sound cue, or simulation did not match the user's expectations. Understanding the factors that contribute to the user's sense of involvement and engagement in an environment is necessary to understand how to build better environments. However, this is a complex problem, but applications which seek to maximize real-world correspondence are likely to be better at meeting users' expectations.

Many virtual environment toolkits such as Deva [PCMW00] Dive [Hag96], Massive [BBFG94], NPSNET [MZP*94], and have exploited human perception to varying degrees to maintain and manage the complexity models representing the virtual environment, and exchange information between participants sharing an environment. If a simulation occurs in the virtual environment but there is no-one to view it then there is little point computing the outcome until it is needed. Similarly, if the effect a user has on a simulated environment exceeds the scope within which they will be aware of it, then there is little point exceeding the perceptible region of influence. We refer interested readers to the survey by Pettifer [Pet99] for a detailed discussion on the exploitation of perception in a number of world models employed by virtual environment toolkits.

However, human perception can be exploited to a far greater degree than to simplify the scope and influence of the user's actions in an environment. A recurring theme linking the algorithms and approaches presented in these course notes is that many exploit human perception to simplify complexity. A better understanding of the way in which humans view images, perform tasks, listen and touch can potentially lead to techniques and algorithms suitable for visual and multi-modal presentation. There is still much research to be done in this area but the techniques we present suggest the potential for compelling realism in real-time.

Part II

Rendering Fidelity for Virtual Environments

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 behavior is crucial to the application.

5 High-Fidelity Graphics

High-fidelity computer graphics aims to produce computer generated images which faithfully represent the real scene they are intended to portray. High-fidelity goes beyond so called, photo-realism, that is images that resemble a photograph. Humans are very adept at identifying photographs and to approximate a photograph such a virtual environment should include blurring and Gaussian noise, which are not present in the real world [LLC03].

Despite advances in rendering techniques it is still not possible to produce images that are exact representations of real environments [UWP05]. Reasons for this include the difficulty in precisely modeling a complex real environment and the limited illumination levels that can be displayed on current display devices. Additionally, artifacts, such as aliasing, can arise during the rendering process itself. This section investigates some of the methods which may be used to determine just how "real" a computer image is.

5.1 Fidelity 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 recognized 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.

5.1.1 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 generalize 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 center 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 normalized by the mean luminance. The non linearity in perception is accounted for by taking the cubed root of each normalized 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 normalized by dividing by the mean luminance. A 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 normalization 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 realized 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.

5.1.2 Low-Level Perception-Based Error Metrics

Perceptual error metrics have also been used in several other areas. Gibson and Hubbold [GH97b] proposed a perceptually-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 analyzed by Myszkowski [Mys98], Gibson et al. [GCHH03] 28, and Volevich et al. [VMKK00].

5.1.3 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

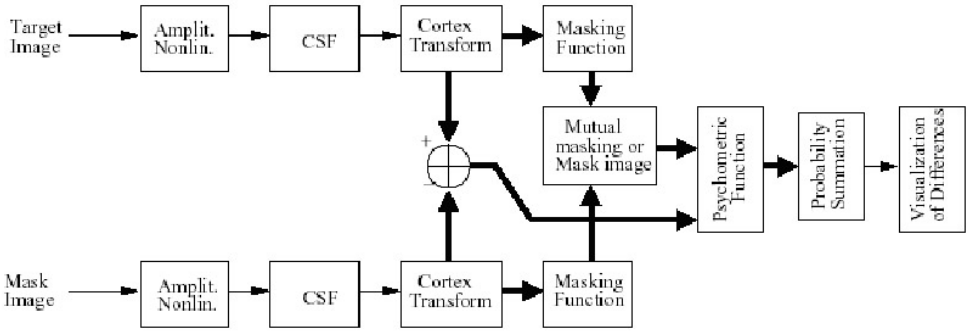


Figure 9: Block diagram of the Visible Differences Predictor (heavy arrows indicate parallel processing of the spatial frequency and orientation channels)

(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 canceled 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.

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 behavioral 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 favorable 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 characterize perceptibility of the differences. The input target and mask images undergo an identical initial processing, as shown in Figure 9. At first, the original pixel intensities are compressed by the amplitude non-linearity based on the local luminance adaptation, simulating Weber's law-like behavior. 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 normalize the contrast differences between target and mask images. The normalized 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 visualize 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 characterize 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

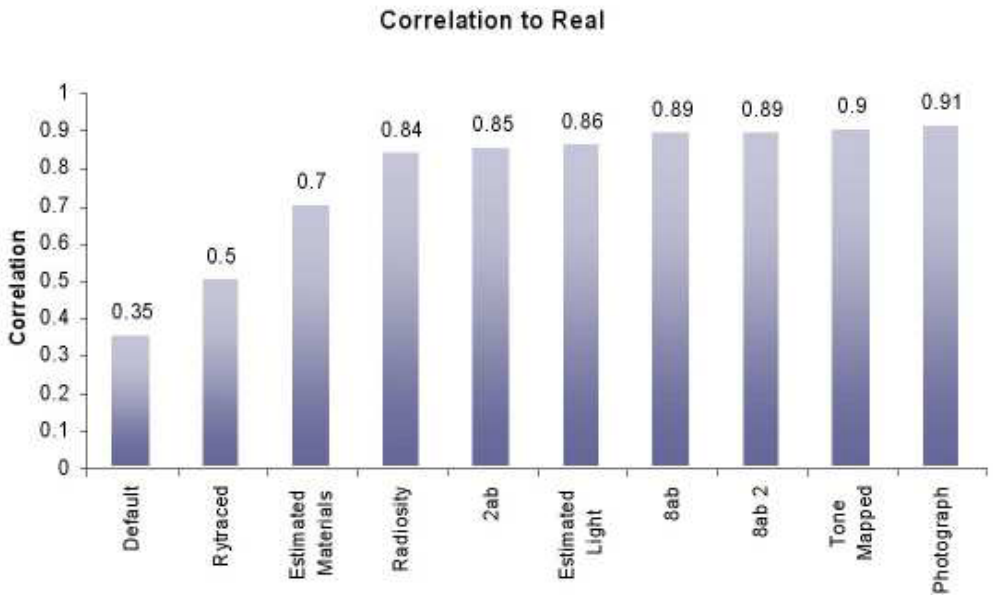


Figure 10: Results obtained by McNamara et al. in lightness matching task experiments

or the quality of shadow reconstruction can be relatively 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.

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 10. In all cases (apart from the ray tracing and radiosity results) Radiance was used to render the images.

Part III

Parallel Processing

Parallel processing is like a dog's walking on its hind legs. It is not done well, but you are surprised to find it done at all.

[Steve Fiddes (University of Bristol) with apologies to Samuel Johnson]

Realistic computer graphics is an area of research which develops algorithms and methods to render images of artificial models or worlds as realistically as possible. Such algorithms are known for their unpredictable data accesses and their high computational complexity. Rendering a single high quality image may take several hours, or even days. Parallel processing offers the potential for solving such complex problems in reasonable times.

However, there are a number of fundamental issues: task scheduling, data management and caching techniques, which must be addressed if parallel processing is to achieve the desired performance when computing realistic images. These are applicable for all global illumination techniques.

This section introduces the concepts of parallel processing, describes its development and considers the difficulties associated with solving problems in parallel. Many further details on this topic may be found in [CDR02].

Parallel processing is an integral part of everyday life. The concept is so ingrained in our existence that we benefit from it without realizing. When faced with a difficult problem, we ask others to help us solve it more easily. This co-operation of more than one worker to facilitate the solution of a particular problem may be termed parallel processing. The goal of parallel processing is thus to solve a given problem more rapidly, or to enable the solution of a problem that would otherwise be impracticable by a single worker.

The concept of parallel processing is not new. Over 2000 years ago the Greeks used such principles in their computational devices. In the Nineteenth Century,

Babbage used parallel processing in order to improve the performance of his Analytical Engine [MA61]. Indeed, the first general purpose electronic digital computer, the ENIAC, was conceived as a highly parallel and decentralized machine with twenty-five independent computing units, co-operating towards the solution of a single problem [Har46]. Two significant perceived problems with parallel processing restricted its widespread acceptance: the complexity of construction; and, the seemingly high programming effort required [Bur81]. These perceived obstacles, together with the increasing availability of seemingly simpler sequential machines led to significant shift in development to optimized sequential algorithms and techniques to the detriment of parallel designs.

The performance of serial computers, however, are limited due to their physical implementation and inherent bottlenecks [Bac78]. As users continue to demand improved performance, computer designers have been looking increasingly at parallel approaches to overcome these limitations. All modern computer architectures incorporate a degree of parallelism. Improved hardware design and manufacture coupled with a growing understanding of how to tackle the difficulties of parallel programming has re-established parallel processing at the forefront of computer technology.

6 Concepts

Parallel processing is the solution of a single problem by dividing it into a number of sub-problems, each of which may be solved by a separate worker. Co-operation will always be necessary between workers during problem solution, even if this is a simple agreement on the division of labor. A simple illustration of this is the solution of the problem of filling a swimming pool with water using buckets. The problem may be sub-divided into the repeated *task* of adding one bucket of water at a time.

A single person completing all the tasks will complete the job, in a certain time. This process may be speed-ed-up by utilizing additional workers. Ideally, two people should be able to fill the pool in half the time. Extending this argument, a large number of workers should be able to complete the job in a small fraction of the original time. However, practically there are physical limitations preventing this hypothetical situation.

Using more than one worker to fill the pool will require a basic level of co- operation between multiple workers. This co-operation is necessary to minimize contention for access to the pool, and thus avoid any collision between the workers. The time required to achieve this co-operation involves inter-worker communication which detracts from the overall solution time, and as such may be termed an *overhead*.

6.1 Dependencies

Dependencies inherent in the problem being tackled may also play a significant role in preventing an ideal parallel solution. Consider the problem of constructing a house. In simple terms, building the roof can only start after the walls have been completed. Similarly, the walls can only be erected once the foundations are laid. The roof is thus dependent upon the walls, which are in turn dependent on the foundations. These dependencies have the effect of dividing the whole problem into a number of distinct stages. The parallel solution of each stage must be completed before the subsequent stage can start.

The dependencies within a problem may be so severe that it is not amenable to parallel processing. A strictly sequential problem consists of a number of stages, each comprising a single task, and each dependent upon the previous stage. For example, in Figure 11, building a tower of toy blocks requires a strictly sequential order of task completion. The situation is the antithesis of dependency-free problems, such as placing blocks in a row on the floor. In this case, the order of task completion is unimportant, but the need for co-operation will still exist.

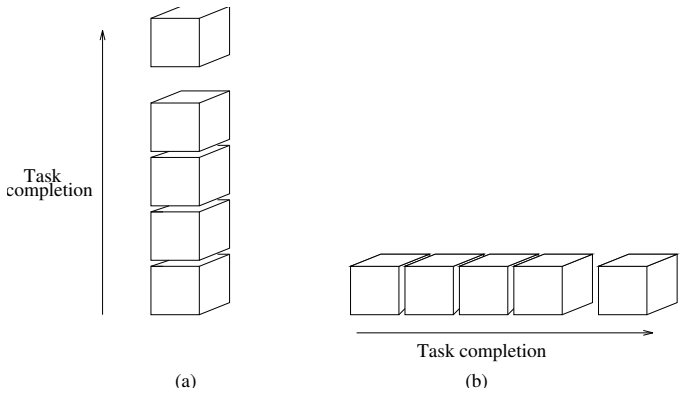


Figure 11: Building with blocks: (a) Strictly sequential (b) dependency-free

Pipelining is the classic methodology for minimizing the effects of dependencies. This technique can only be exploited when a process, consisting of a number of distinct stages, needs to be repeated several times. An automotive assembly line is an example of an efficient

pipeline. In a simplistic form, the construction of a car may consist of four linearly dependent stages: chassis fabrication; body assembly; wheel fitting; and, windscreen installation. An initial lump of metal is introduced into the pipeline then, as the partially completed car passes each stage, a new section is added until finally the finished car is available outside the factory.

Consider an implementation of this process consisting of four workers, each performing their task in one time unit. Having completed the task, the worker passes the partially completed car on to the next stage. This worker is now free to repeat its task on a new component fed from the previous stage. The completion of the first car occurs after four time units, but each subsequent car is completed every time unit.

The completion of a car is, of course, sensitive to the time taken by each worker. If one worker were to take longer than one time unit to complete its task then the workers after this difficult task would stand idle awaiting the next component, whilst those before the worker with the difficult task would be unable to move their component on to the next stage of the pipeline. The other workers would thus also be unable to do any further work until the difficult task was completed. Should there be any interruption in the input to the pipeline then the pipeline would once more have to be "refilled" before it could operate at maximum efficiency.

6.2 Scalability

Every problem contains a upper bound on the number of workers which can be meaningfully employed in its solution. Additional workers beyond this number will not improve solution time, and can indeed be detrimental. This upper bound provides an idea as to how suitable a problem is to parallel implementation: a measure of its *scalability*.

A given problem may only be divided into a finite number of sub-problems, corresponding to the smallest tasks. The availability of more workers than there are tasks, will not improve solution time. The problem of clearing a room of 100 chairs may be divided into 100 tasks consisting of removing a single chair. A maximum of 100 workers can be allocated one of these tasks and hence perform useful work.

The optimum solution time for clearing the room may not in fact occur when employing 100 workers due to certain aspects of the problem limiting effective worker utilization. This phenomenon can be illustrated by adding a constraint to the problem, in the form of a single doorway providing egress from the room. A *bottleneck* will occur as large numbers of workers attempt to move their chairs through the door simultaneously, as shown in Figure 12.

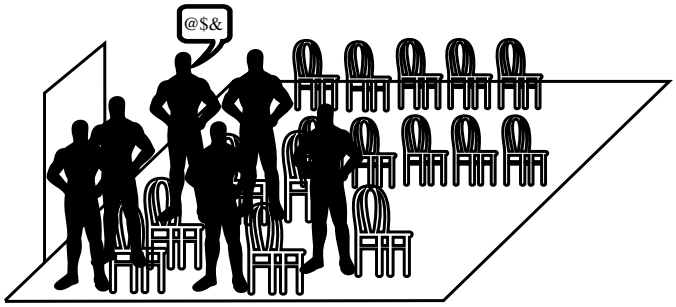


Figure 12: Bottleneck caused by doorway

The delays caused by this bottleneck may be so great that the time taken to empty the room of chairs by this large number of workers may in fact be *longer* than the original time taken by the single worker. In this case, reducing the number of workers can alleviate the bottleneck and thus reduce solution time.

6.3 Control

All parallel solutions of a problem require some form of control. This may be as simple as the control needed to determine what will constitute a task and to ascertain when the problem has been solved satisfactorily. More complex problems may require control at several stages of their solution. For example, solution time could be improved when clearing the room if a controller was placed at the door to schedule its usage. This control would ensure that no time was wasted by two (or more) workers attempting to exit simultaneously and then having to "reverse" to allow a single worker through. An alternative to this explicit *centralized* control would be some form of *distributed* control. Here the workers themselves could have a way of preventing simultaneous access, for example, if two (or more) workers reach the door at the same time then the biggest worker will always go first while the others wait.

Figure 13(a) shows the sequential approach to solving a problem. Computation is applied to the problem domain to produce the desired results. The controlled parallel approach shown in Figure 13(b) achieves a parallel implementation of the same problem via three steps. In step 1, the problem domain is divided into a number of sub-problems, in this case four. Parallel processing is introduced in step 2 to enable each of the sub-problems to be computed in parallel to produce sub-results. In step 3, these results must now be collated to achieve the desired final results. Control is necessary in steps 1 and 3 to divide the problem amongst the workers and then to collect and collate the results that the workers have independently produced.

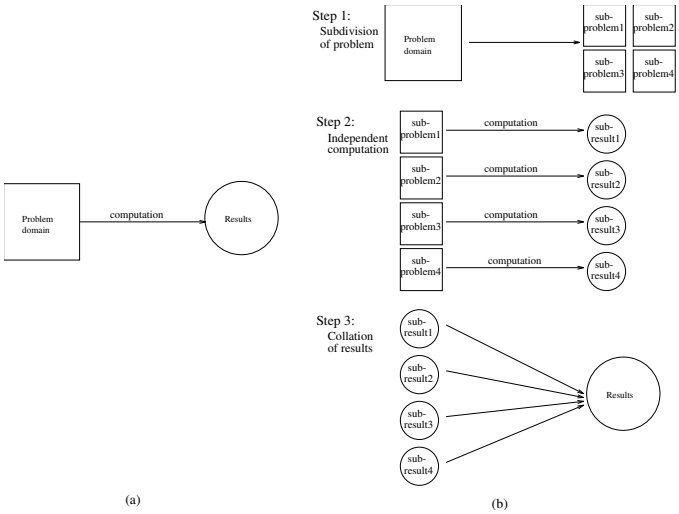


Figure 13: Control required in (a) a Sequential versus (b) a parallel implementation

7 The Relationship of Tasks and Data

The implementation of any problem on a computer comprises two components:

- the algorithm chosen to solve the problem; and,
- the domain of the problem which encompasses all the data requirements for that problem.

The algorithm interacts with the domain to produce the result for the problem, as shown diagrammatically in Figure 14.

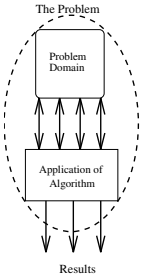


Figure 14: The components of a problem

A sequential implementation of the problem means that the entire algorithm and domain reside on a single processor. To achieve a parallel implementation it is necessary to divide the problem's components in some manner amongst the parallel processors. Now no longer resident on a single processor, the components will have to interact within the multiprocessor system in order to obtain the same result. This co-operation requirement introduces a number of novel difficulties into any parallel implementation which are not present in the sequential version of the same problem.

7.1 Inherent Difficulties

User confidence in any computer implementation of a problem is bolstered by the successful termination of the computation and the fact that the results meet design specifications. The reliability of modern computer architectures and languages is such that any failure of a sequential implementation to complete successfully will point automatically to deficiencies in either the algorithm used or data supplied to the program. In addition to these possibilities of failure, a parallel implementation may also be affected by a number of other factors which arise from the manner of the implementation:

Deadlock: An active parallel processor is said to be deadlocked if it is waiting indefinitely for an event which will never occur. A simple example of deadlock is when two processors, using synchronized communication, attempt to send a message to each other at the same time. Each process will then wait for the other process to perform the corresponding input operation which will never occur.

Data consistency: In a parallel implementation, the problem's data may be distributed across several processors. Care has to be taken to ensure:

- if multiple copies of the same data item exists then the value of this item is kept consistent;
- mutual exclusion is maintained to avoid several processors accessing a shared resource simultaneously; and,
- the data items are fetched from remote locations efficiently in order to avoid processor idle time.

While there is meaningful computation to be performed, a sequential computer is able to devote 100% of its time for this purpose. In a parallel system it may happen that some of the processors become idle, not because there is no more work to be done, but because current circumstances prevent those processors being able to perform any computation. Parallel processing introduces communication overheads. The effect of these overheads is to introduce latency into the multiprocessor system. Unless some way is found to minimize communication delays, the percentage of time that a processor can spend on useful computation may be significantly affected. So, as well as the factors affecting the successful termination of the parallel implementation, one of the fundamental considerations also facing parallel programmers is the *computation to communication* ratio.

7.2 Tasks

Subdividing a single problem amongst many processors introduces the notion of a task. In its most general sense, a task is a unit of computation which is assigned to a processor within the parallel system. In any parallel implementation a decision has to be taken as to what exactly constitutes a task. The *task granularity* of a problem is a measure of the amount of computational effort associated with any task. The choice of granularity has a direct bearing on the computation to communication ratio. Selection of too large a granularity may prevent the solution of the problem on a large parallel system, while too fine a granularity may result in significant processor idle time while the system attempts to keep processors supplied with fresh tasks. On completion of a sequential implementation of a problem, any statistics that may have been gathered during the course of the computation, may now be displayed in a straightforward manner. Furthermore, the computer is in a state ready to commence the next sequential program. In a multiprocessor system, the statistics would have been gathered at each processor, so after the solution of the problem the programmer is still faced with the task of collecting and collating these statistics. To ensure that the multiprocessor system is in the correct state for the next parallel program, the programmer must also ensure that all the processors have *terminated gracefully*.

7.3 Data

The problem domains of many rendering applications are very large. The size of these domains are typically far more than can be accommodated within the local memory of any processing element (or indeed in the memory of many sequential computers). Yet it is precisely these complex problems that we wish to solve using parallel processing. Consider a multiprocessor system consisting of sixty-four processing elements each with 4 MBytes of local memory. If we were to insist that the entire problem domain were to reside at each processing element then we would be restricted to solving problems with a maximum domain of 4 MBytes. The total memory within the system is $64 \times 4 = 256$ MBytes. So, if we were to consider the memory of the multiprocessor system as a whole, then we could contemplate solving problems with domains of up to 256 MBytes in size; a far more attractive proposition. (If the problem domain was even larger than this, then we could also consider the secondary storage devices as part of the combined memory and that should be sufficient for most problems.) There is a price to pay in treating the combined memory as a single unit. Data management strategies will be necessary to translate between the conceptual single memory unit and the physical distributed implementation. The aims of these strategies will be to keep track of the data items so that an item will always be available at a processing element when required by the task being performed. The distributed nature of the data items will thus be invisible to the application processes performing the computation. However, any delay between the application process requesting an item and this request being satisfied will result in idle time. As we will see, it is the responsibility of data management to avoid this idle time.

7.4 Evaluating Parallel Implementations

The chief reason for opting for a parallel implementation should be: *to obtain answers faster*. The time that the parallel implementation takes to compute results is perhaps the most natural way of determining the benefits of the approach that has been taken. If the parallel solution takes *longer* than any sequential implementation then the decision to use parallel processing needs to be re-examined. Other measurements, such as speed-up and efficiency, may also provide useful insight on the maximum scalability of the implementation. Of course, there are many issues that need to be considered when comparing parallel and sequential implementations of the same problem, for example:

- Was the same processor used in each case?
- If not, what is the price of the sequential machine compared with that of the multiprocessor system?
- Was the algorithm chosen already optimized for sequential use, that is, did the data dependencies present preclude an efficient parallel implementation?

7.4.1 Realization Penalties

If we assume that the same processor was used in both the sequential and parallel implementation, then we should expect,, that the time to solve the problem decreases as more processing elements are added. The best we can reasonably hope for is that two processing elements will

solve the problem twice as quickly, three processing elements three times faster, and *n* processing elements, *n* times faster. If *n* is sufficiently large then by this process, we should expect our large scale parallel implementation to produce the answer in a tiny fraction of the sequential computation, as shown by the "optimum time" curve in the graph in Figure 15.

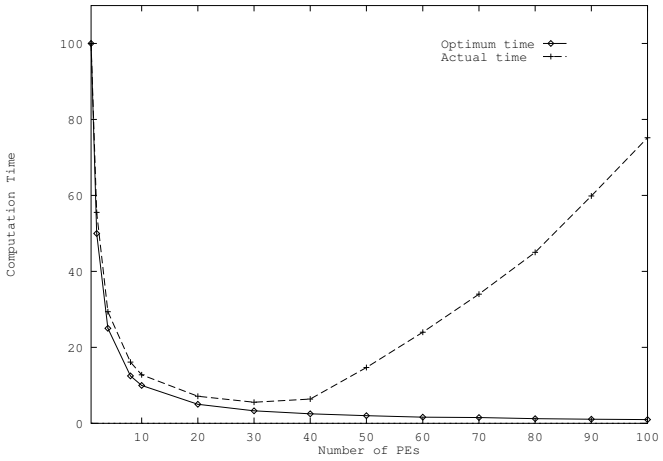


Figure 15: Optimum and actual parallel implementation times

However, in reality we are unlikely to achieve these optimized times as the number of processors is increased. A more realistic scenario is that shown by the curve "actual times" in Figure 15. This curve shows an initial decrease in time taken to solve the example problem on the parallel system up to a certain number of processing elements. Beyond this point, adding more processors actually leads to an *increase* in computation time. Failure to achieve the optimum solution time means that the parallel solution has suffered some form of *realization* penalty. A realization penalty can arise from two sources:

- an *algorithmic* penalty; and,
- an *implementation* penalty.

The algorithmic penalty stems from the very nature of the algorithm selected for parallel processing. The more inherently sequential the algorithm, the less likely the algorithm will be a good candidate for parallel processing. **Aside:** It has also been shown, albeit not conclusively, that the more experience the writer of the parallel algorithm has in sequential algorithms, the less parallelism that algorithm is likely to exhibit [Cha88]. This sequential nature of an algorithm and its implicit data dependencies will translate, in the domain decomposition approach, to a requirement to *synchronize* the processing elements at certain points in the algorithm. This can result in processing elements standing idle awaiting messages from other processing elements. A further algorithmic penalty may also come about from the need to reconstruct sequentially the results generated by the individual processors into an overall result for the computation. Solving the same problem twice as fast on two processing elements implies that those two processing elements must spend 100% of their time on computation. We know that a parallel implementation requires some form of communication. The time a processing element is forced to spend on communication will naturally impinge on the time a processor has for computation. Any time that a processor cannot spend doing useful computation is an implementation penalty. Implementation penalties are thus caused by:

- **the need to communicate** As mentioned above, in a multiprocessor system, processing elements need to communicate. This communication may not only be that which is necessary for a processing element's own actions, but in some architectures, a processing element may also have to act as a intermediate for other processing elements' communication.
- **idle time** Idle time is any period of time when an application process is available to perform some useful computation, but is unable to do so because either there is no work locally available, or its current task is suspended awaiting a synchronization signal, or a data item which has yet to arrive. It is the job of the local task manager to ensure that an application process is kept supplied with work. The computation to communication ratio within the system will determine how much time a task manager has to fetch a task before the current one is completed. A *load imbalance* is said to exist if some processing elements still have tasks to complete, while the others do not. While synchronization points are introduced by the algorithm, the management of data items for a processing element is the job for the local data manager. The domain decomposition approach means that the problem domain is divided amongst the processing elements in some fashion. If an application process requires a data item that is not available locally, then this must be fetched from some other processing element within the system. If the processing element is unable to perform other useful computation while this fetch is being performed, for example by means of multi-threading, then the processing element is said to be idle
- **concurrent communication, data management and task management activity** Implementing each of a processing element's activities as a separate concurrent process on the same processor, means that the physical processor has to be shared. When another process other than the application process is scheduled then the processing element is not performing useful computation even though its current activity is necessary for the parallel implementation.

The fundamental goal of the system software is to minimize the implementation penalty. While this penalty can never be removed, intelligent communication, data management and task scheduling strategies can avoid idle time and significantly reduce the impact of the need to

communicate.

7.4.2 Performance Metrics

Solution time provides a simple way of evaluating a parallel implementation. However, if we wish to investigate the relative merits of our implementation then further insight can be gained by additional metrics. A range of metrics will allow us to compare aspects of different implementations and perhaps provide clues as to how overall system performance may be improved.

Speed-up

A useful measure of any multiprocessor implementation of a problem is *speed-up*. This relates the time taken to solve the problem on a single processor machine to the time taken to solve the same problem using the parallel implementation. We will define the speed-up of a multiprocessor system in terms of the elapsed time that is taken to complete a given problem, as follows:

$$\text{Speed-up} = \frac{\text{elapsed time of a uniprocessor}}{\text{elapsed time of the multiprocessors}} \quad (1)$$

The term *linear speed-up* is used when the solution time on an n processor system is n times faster than the solution time on the uniprocessor. This linear speed-up is thus equivalent to the optimum time shown in section 7.4.1. The optimum and actual computation times in Figure 15 are represented as a graph of linear and actual speed-ups in Figure 16. Note that the actual speed-up curve increases until a certain point and then subsequently decreases. Beyond this point we say that the parallel implementation has suffered a *speed-down*.

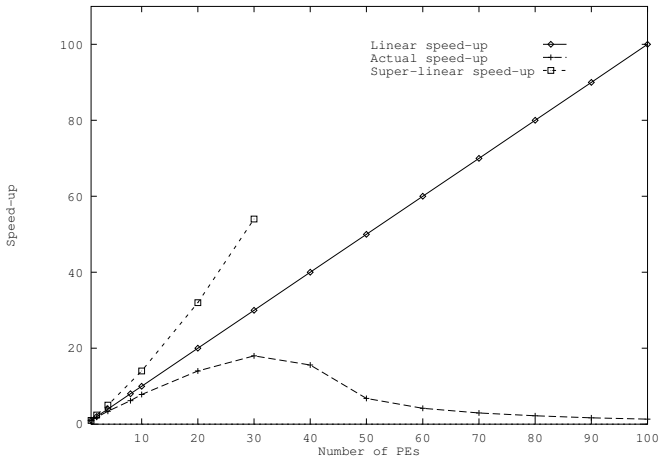


Figure 16: Linear and actual speed-ups

The third curve in Figure 16 represents so-called *super-linear speed-up*. In this example, the implementation on 20 processors has achieved a computation time which is approximately 32 times faster than the uniprocessor solution. It has been argued that it is not possible to achieve a speed-up greater than the number of processors used. While in practice it certainly is possible to achieve super-linear speed-up, such implementation may have exploited "unfair" circumstances to obtain such timings. For example, most modern processors have a limited amount of cache memory with an access time significantly faster compared with a standard memory access. Two processors would have double the amount of this cache memory. Given we are investigating a fixed size problem, this means that a larger proportion of the problem domain is in the cache in the parallel implementation than in the sequential implementation. It is not unreasonable, therefore, to imagine a situation where the two processor solution time is more than twice as fast than the uniprocessor time.

Although super-linear speed-up is desirable, in this tutorial we will assume a "fair" comparison between uniprocessor and multiprocessor implementations. The results that are presented in the case studies thus make no attempt to exploit any hardware advantages offered by the increasing number of processors. This will enable the performance improvements offered by the proposed system software extensions to be highlighted without being masked by any variations in underlying hardware. In practice, of course, it would be foolish to ignore these benefits and readers are encouraged to "squeeze every last ounce of performance" out of their parallel implementation.

Two possibilities exist for determining the "elapsed time of a uniprocessor". This could be the time obtained when executing:

1. an optimized sequential algorithm on a single processor, T_s; or,
2. the parallel implementation on *one* processing element, T_1.

The time taken to solve the problem on n processing elements we will term T_n. The difference between how the two sequential times are obtained is shown in Figure 17. There are advantages in acquiring both these sequential times. Comparing the parallel to the optimized

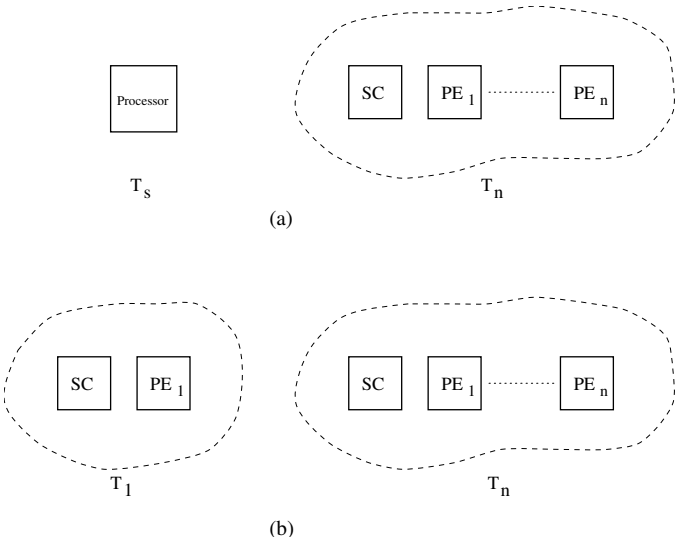


Figure 17: Systems used to obtain T_n and (a) T_s (b) T_1

sequential implementation highlights any algorithmic efficiencies that had to be sacrificed to achieve the parallel version. In addition, none of the parallel implementation penalties are hidden by this comparison and thus the speed-up is not exaggerated. One of these penalties is the time taken simply to supply the data to the processing element and collect the results. The comparison of the single processing element with the multiple processing element implementation shows how well the problem is "coping" with an increasing number of processing elements. Speed-up calculated as $\frac{T_1}{T_n}$, therefore, provides the indication as to the *scalability* of the parallel implementation. Unless otherwise stated, we will use this alternative for speed-up in the case studies in this book as it better emphasizes the performance improvements brought about by the system software we shall be introducing. As we can see from the curve for "actual speed-up" in Figure 16, the speed-up obtained for that problem increased to a maximum value and then subsequently decreased as more processing elements were added. In 1967 Amdahl presented what has become known as "Amdahl's law" [Amd67]. This "law" attempts to give a maximum bound for speed-up from the nature of the algorithm chosen for the parallel implementation. We are given an algorithm in which the proportion of time that needs to be spent on the purely sequential parts is s, and the proportion of time that might be done in parallel is p, by definition. The total time for the algorithm on a single processor is $s+p=1$ (where the 1 is for algebraic simplicity), and the maximum speed-up that can be achieved on n processors is:

$$\begin{aligned} \text{maximum speed-up} &= \frac{(s+p)}{s+\frac{p}{n}} \\ &= \frac{1}{s+\frac{p}{n}} \end{aligned} \tag{2}$$

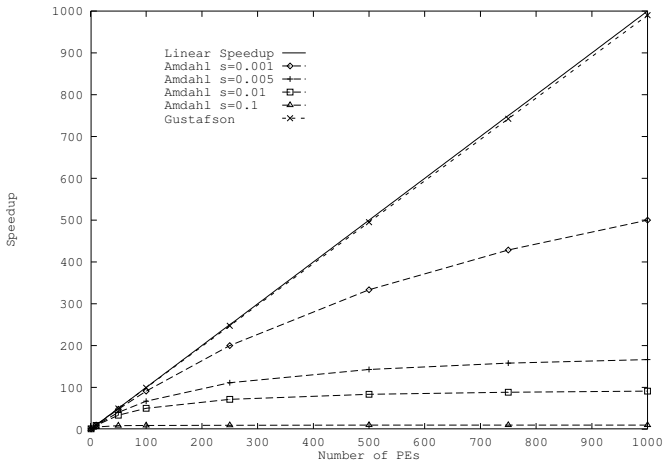


Figure 18: Example maximum speed-up from Amdahl and Gustafson's laws

Figure 18 shows the maximum speed-up predicted by Amdahl's law for a sequential portion of an algorithm requiring 0.1%, 0.5%, 1% and 10% of the total algorithm time, that is $s = 0.001,\ 0.005,\ 0.01$ and 0.1 respectively. For 1000 processors the maximum speed-up that can be achieved for a sequential portion of only 1% is less than 91. This rather depressing forecast put a serious damper on the possibilities of massive parallel implementations of algorithms and led Gustafson in 1988 to issue a counter claim [Gus88]. Gustafson stated that a problem size is virtually never independent of the number of processors, as it appears in equation (2), but rather:

> *... in practice, the problem size scales with the number of processors.*

Gustafson thus derives a maximum speed-up of:

$$\begin{aligned} \text{maximum speed-up} &= \frac{(s+(p\times n))}{s+p} \\ &= n+(1-n)\times s \end{aligned} \tag{3}$$

This maximum speed-up according to Gustafson is also shown in Figure 18. As the curve shows, the maximum achievable speed-up is nearly linear when the problem size is increased as more processing elements are added. Despite this optimistic forecast, Gustafson's premise is not applicable in a large number of cases. Most scientists and engineers have a particular problem they want to solve in as short a time as possible. Typically, the application already has a specified size for the problem domain. For example, in parallel radiosity we will be considering the diffuse lighting within a particular environment subdivided into a necessary number of patches. In this example it would be inappropriate for us to follow Gustafson's advice and increase the problem size as more processing elements were added to their parallel implementation, because to do so would mean either:

- the physical size of the three dimensional objects within the environment would have to be increased, which is of course not possible; or,
- the size of the patches used to approximate the surface would have to be reduced, thereby increasing the number of patches and thus the size of the problem domain.

This latter case is also not an option, because the computational method is sensitive to the size of the patches relative to their distances apart. Artificially significantly decreasing the size of the patches may introduce numerical instabilities into the method. Furthermore, artificially increasing the size of the problem domain may improve speed-up, but it *will not* improve the time taken to solve the problem. For fixed sized problems it appears that we are left with Amdahl's gloomy prediction of the maximum speed-up that is possible for our parallel implementation. However, all is not lost, as Amdahl's assumption that an algorithm can be separated into a component which has to be executed sequentially and part which can be performed in parallel, may not be totally appropriate for the domain decomposition approach. Remember, in this model we are retaining the complete sequential algorithm and exploiting the parallelism that exists in the problem domain. So, in this case, an equivalent to Amdahl's law would imply that the data can be divided into two parts, that which must be dealt with in a strictly sequential manner and that which can executed in parallel. Any data dependencies will certainly imply some form of sequential ordering when dealing with the data, however, for a large number of problems such data dependencies may not exist. It may also be possible to reduce the effect of dependencies by clever scheduling. The achievable speed-up for a problem using the domain decomposition approach is, however, bounded by the number of tasks that make up the problem. Solving a problem comprising a maximum of twenty tasks on more than twenty processors makes no sense. In practice, of course, any parallel implementation suffers from realization penalties which increase as more processing elements are added. The actual speed-up obtained will thus be less than the maximum possible speed- up.

7.4.3 Efficiency

A relative efficiency based on the performance of the problem on one processor, can be a useful measure as to what percentage of a processor's time is being spent in useful computation. This, therefore, determines what the system overheads are. The relative efficiency we will measure as:

$$\text{Efficiency} = \frac{\text{speed-up}\times 100}{\text{number of processors}} \tag{4}$$

Figure 19 shows the optimum and actual computation times given in Figure 15 represented as processing element efficiency. The graph shows that optimum computation time, and therefore linear speed-up, equates to an efficiency of 100% for each processing element. This again shows that to achieve this level of efficiency every processing element must spend 100% of its time performing useful computation. Any implementation penalty would be immediately reflected by a decrease in efficiency. This is clearly shown in the curve for the actual computation times. Here the efficiency of each processing element decreases steadily as more are added until by the time 100 processing elements are incorporated, the realization penalties are so high that each processing element is only able to devote just over 1% of its time to useful computation.

Optimum Number of Processing Elements

Faced with implementing a fixed size problem on a parallel system, it may be useful to know the optimum number of processing elements on which this particular problem should be implemented in order to achieve the best possible performance. We term this optimum number n_{opt}.

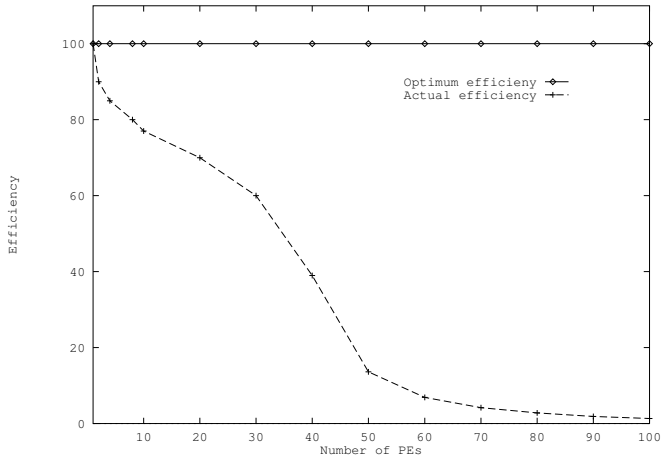


Figure 19: Optimum and actual processing element efficiency

We shall judge the *maximum performance* for a particular problem with a fixed problem domain size, as the shortest possible time required to produce the desired results for a certain parallel implementation. This optimum number of processing elements may be derived directly from the "computation time" graph. In Figure 15 the minimum actual computation time occurred when the problem was implemented on 30 processing elements. As Figure 20 shows, this optimum number of processing elements is also the point on the horizontal axis in Figure 16 at which the maximum speed-up was obtained.

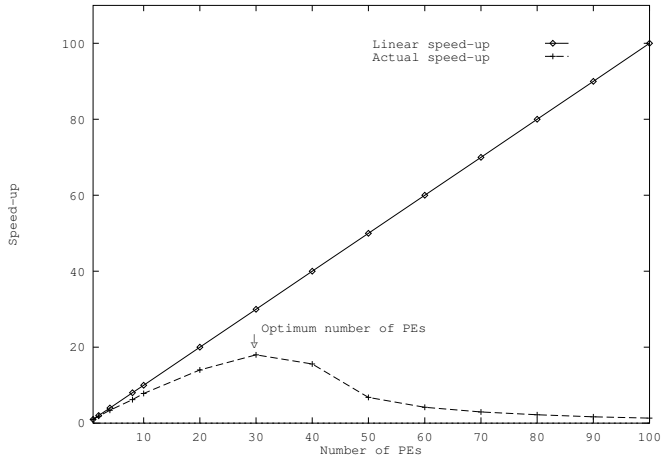


Figure 20: Optimum number of processing elements related to speed-up

The optimum number of processing elements is also the upper bound for the scalability of the problem for that parallel implementation. To improve the scalability of the problem it is necessary to re-examine the decisions concerning the algorithm chosen and the make-up of the system software that has been adopted for supporting the parallel implementation. As we will see in the subsequent chapters, the correct choice of system software can have a significant effect on the performance of a parallel implementation.

Figure 21 shows the speed-up graphs for different system software decisions for the *same* problem. The goal of a parallel implementation may be restated as:

> *"to ensure that the optimum number of processing elements for your problem is greater than the number of processing elements physically available to solve the problem!"*

Other Metrics

Computation time, speed-up and efficiency provide insight into how successful a parallel implementation of a problem has been. As Figure 21 shows, different implementations of the same algorithm on the same multiprocessor system may produce very different performances. A multitude of other metrics have been proposed over the years as a means of comparing the relative merits of different architectures and to provide a way of assessing their suitability as the chosen multiprocessor machine. The performance of a computer is frequently measured as the rate of some number of events per second. Within a multi-user environment the elapsed time to solve a problem will comprise the user's

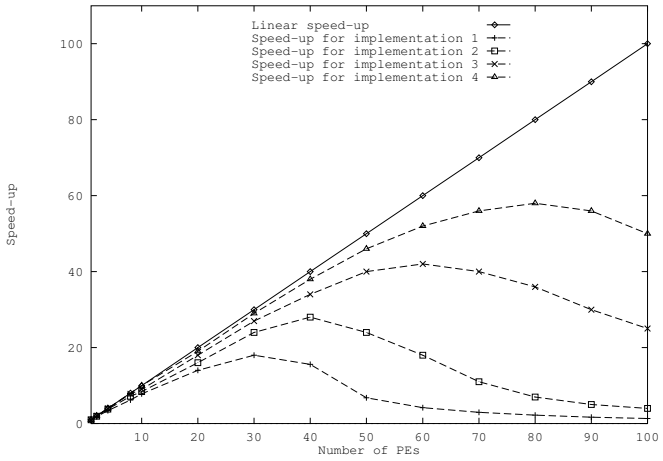


Figure 21: Speed-up graphs for different system software for the same problem

CPU time plus the system's CPU time. Assuming that the computer's clock is running at a constant rate, the user's CPU performance may be measured as:

$$\text{CPU time} = \frac{\text{CPU clock cycles for a program}}{\text{clock rate (eg. 100MHz)}}$$

The average clock cycles per instruction (CPI) may be calculated as:

$$\text{CPI} = \frac{\text{CPU clock cycles for a program}}{\text{Instruction count}}$$

We can also compute the CPU time from the time a program took to run:

$$\begin{aligned} \text{CPU time} &= \frac{\text{seconds}}{\text{program}} \\ &= \frac{\text{seconds}}{\text{clock cycle}} \times \frac{\text{clock cycles}}{\text{instructions}} \times \frac{\text{instructions}}{\text{program}} \end{aligned}$$

Such a performance metric is dependent on:

Clock rate: this is determined by the hardware technology and the organization of the architecture;

CPI: a function of the system organization and the instruction set architecture; and,

Instruction count: this is affected by the instruction set architecture and the compiler technology utilized.

One of the most frequently used performance metrics is the *MIPS* rating of a computer, that is how many Million Instructions Per Second the computer is capable of performing:

$$\text{MIPS} = \frac{\text{instruction count}}{\text{execution time} \times 10^6} = \frac{\text{clock rate}}{\text{CPI} \times 10^6}$$

However, the MIPS value is dependent on the instruction set used and thus any comparison between computers with different instruction sets is not valid. The MIPS value may even vary between programs running on the same computer. Furthermore, a program which makes use of hardware floating point routines may take *less time* to complete than a similar program which uses a software floating point implementation, but the first program will have a *lower* MIPS rating than the second. These anomalies have led to MIPS sometimes being referred to as *"Meaningless Indication of Processor Speed"*. Similar to MIPS is the "Mega-FLOPS" (MFLOPS) rating for computers, where MFLOPS represents Million FLoating point OPerations per Second:

$$\text{MFLOPS} = \frac{\text{no. of floating point operations in a program}}{\text{execution time} \times 10^6}$$

MFLOPS is not universally applicable, for example a word processor utilizing no floating point operations would register no MFLOPS rating. However, the same program executing on different machines should be comparable, because, although the computers may execute a different

number of instructions, they should perform the same number of operations, provided the set of floating point operations is consistent across both architectures. The MFLOPS value will vary for programs running on the same computer which have different mixtures of integer and floating point instructions as well as a different blend of "fast" and "slow" floating point instructions. For example, the *add* instruction often executes in less time than a *divide* instruction. A MFLOPS rating for a single program can not, therefore, be generalized to provide a single performance metric for a computer. A suite of benchmark programs, such as the LINPACK or Livermore Loops routines, have been developed to allow a more meaningful method of comparison between machines. When examining the relative performance of computers using such benchmarks it is important to discover the *sustained* MFLOPS performance as a more accurate indication of the machines' potential rather than merely the *peak* MFLOPS rating, a Figure that *"can be guaranteed never to be exceeded"*. Other metrics for comparing computers include:

Dhrystone: A CPU intensive benchmark used to measure the integer performance especially as it pertains to system programming.

Whetstone: A synthetic benchmark without any factorizable code for evaluating floating point performance.

TPS: Transactions Per Second measure for applications, such as airline reservation systems, which require on-line database transactions.

KLIPS: Kilo Logic Inferences Per Second is used to measure the relative inference performance of artificial intelligence machines

Tables showing the comparison of the results of these metrics for a number of architectures can be found in several books, for example [HJ88]. Cost is seldom an issue that can be ignored when purchasing a high performance computer. The desirability of a particular computer or even the number of processors within a system may be offset by the extraordinarily high costs associated with many high performance architectures. This prompted an early "law" by Grosch that the speed of a computer is proportional to its cost. Fortunately, although this is no longer completely true, multiprocessor machines are nevertheless typically more expensive than their general purpose counterparts. The parallel computer eventually purchased should provide acceptable computation times for an affordable price, that is maximize: *"the bangs per buck"* (performance per unit price).

8 Task Scheduling and Data Management

The efficient solution of a problem on a parallel system requires the computational performance of the processing elements to be fully utilized. Any processing element that is not busy performing useful computations is degrading overall system performance. Task scheduling strategies may be used to minimize these potential performance limitations.

8.1 Problem Decomposition

A problem may be solved on a parallel system by either exploiting the parallelism inherent in the algorithm, known as *algorithmic decomposition*, or by making use of the fact that the algorithm can be applied to different parts of the problem domain in parallel, which is termed *domain decomposition*. These two decomposition methods can be further categorized as shown in Figure 22.

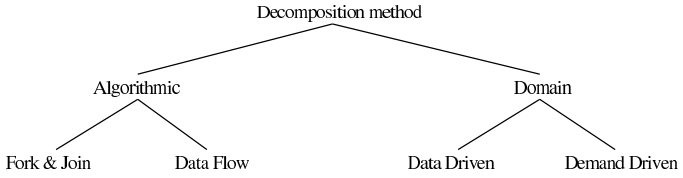


Figure 22: Methods of decomposing a problem to exploit parallelism

Over the years, an abundance of algorithms have been developed to solve a multitude of problems on sequential machines. A great deal of time and effort has been invested in the production of these sequential algorithms. Users are thus loathed to undertake the development of novel parallel algorithms, and yet still demand the performance that multiprocessor machines have to offer.

Algorithmic decomposition approaches to this dilemma have led to the development of compilers, such as those for High Performance Fortran, which attempt to parallelize automatically these existing algorithms. Not only do these compilers have to identify the parallelism hidden in the algorithm, but they also need to decide upon an effective strategy to place the identified segments of code within the multiprocessor system so that they can interact efficiently. This has proved to be an extremely hard goal to accomplish.

The domain decomposition approach, on the other hand, requires little or no modification to the existing sequential algorithm. There is thus no need for sophisticated compiler technology to Analise the algorithm. However, there will be a need for a parallel framework in the form of system software to support the division of the problem domain amongst the parallel processors.

8.1.1 Algorithmic Decomposition

In algorithmic decomposition the algorithm itself is analyzed to identify which of its features are capable of being executed in parallel. The finest granularity of parallelism is achievable at the operation level. Known as *dataflow*, at this level of parallelism the data "flows" between

individual operands which are being executed in parallel. An advantage of this type of decomposition is that little data space is required per processor, however, the communication overheads may be very large due to the very poor computation to communication ratio.

Fork & join parallelism, on the other hand, allocates portions of the algorithm to separate processors as the computation proceeds. These portions are typically several statements or complete procedures. The difference between the two algorithmic forms of decomposition is shown for a simple case in Figure 23.

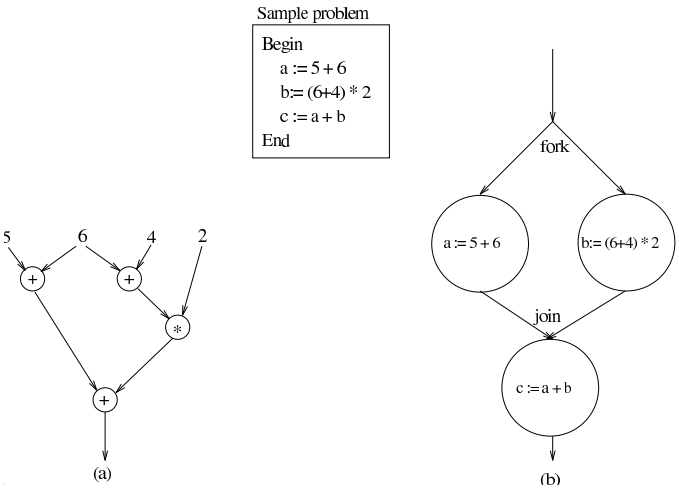


Figure 23: Algorithmic decomposition: (a) dataflow (b) fork & join

8.1.2 Domain Decomposition

Instead of determining the parallelism inherent in the algorithm, domain decomposition examines the problem domain to ascertain the parallelism that may be exploited by solving the algorithm on distinct data items in parallel. Each parallel processor in this approach will, therefore, have a complete copy of the algorithm and it is the problem domain that is divided amongst the processors. Domain decomposition can be accomplished using either a data driven or demand driven approach.

As we shall see, given this framework, the domain decomposition approach is applicable to a wide range of problems. Adoption of this approach to solve a particular problem in parallel, consists of two steps:

1. **Choosing the appropriate sequential algorithm**

   Many algorithms have been honed over a number of years to a high level of perfection for implementation on sequential machines. The data dependencies that these highly sequential algorithms exhibit may substantially inhibit their use in a parallel system. In this case alternative sequential algorithms which are more suitable to the domain decomposition approach will need to be considered.

2. **Analysis of the problem in order to extract the criteria necessary to determine the optimum system software.**

   The system software provides the framework in which the sequential algorithm can execute. This system software takes care of ensuring each processor is kept busy, the data is correctly managed, and any communication within the parallel system is performed rapidly. To provide maximum efficiency, the system software needs to be tailored to the requirements of the problem. There is thus no *general purpose* parallel solution using the domain decomposition approach, but, as we shall see, a straightforward analysis of any problem's parallel requirements, will determine the correct construction of the system software and lead to an efficient parallel implementation.

Before commencing the detailed description of how we intend to tackle the solution of realistic rendering problems in parallel, it might be useful to clarify some of the terminology we shall be using.

8.1.3 Abstract Definition of a Task

The domain decomposition model solves a single problem in parallel by having multiple processors apply the same sequential algorithm to different data items from the problem domain in parallel. The lowest unit of computation within the parallel system is thus the application of the algorithm to one data item within the problem domain.

The data required to solve this unit of computation consists of two parts:

1. the *principal data items* (or PDIs) on which the algorithm is to be applied; and
2. *additional data items* (or ADIs) that may be needed to complete this computation on the PDIs.

For example, in ray tracing, we are computing the value at each pixel of our image plane. Thus these pixels would form our PDIs, while all the data describing the scene would constitute the ADIs. The problem domain is thus the pixels *plus* the scene description.

The application of the algorithm to a specified principal data item may be regarded as performing a single *task*. The task forms the elemental unit of computation within the parallel implementation. This is shown diagrammatically in Figure 24.

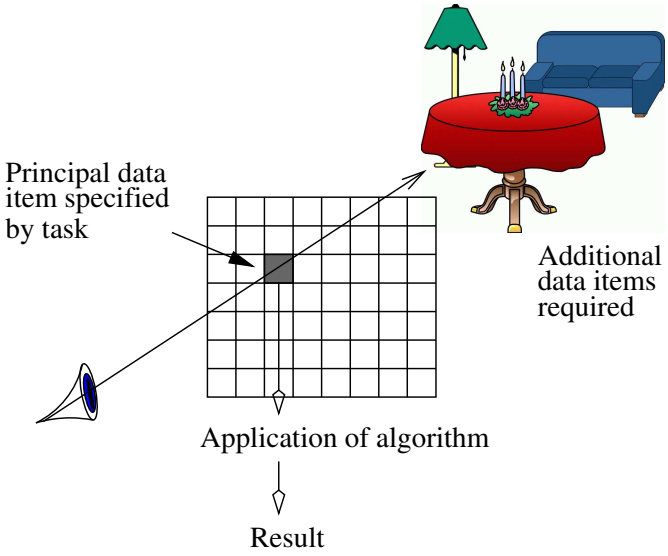


Figure 24: A task: the processing of a *principal* data item

8.2 System Architecture

This section is concentrating on implementing realistic rendering techniques on distributed memory systems (either a dedicated parallel machine or a cluster of workstations). These processors may be connected together in some manner to form a *configuration*. A *process* is a segment of code that runs concurrently with other processes on a single processor. Several processes will be needed at each processor to implement the desired application and provide the necessary system software support. A *processing element* consists of a single processor together with these application and system processes and is thus the building block of the *multiprocessor system*. (We shall sometimes use the abbreviation *PE* for processing element in the figures and code segments.) When discussing configurations of processing elements, we shall use the term *links* to mean the communication paths between processes.

8.3 Computational Models

The computational model chosen to solve a particular problem determines the manner in which work is distributed across the processors of the multiprocessor system. In our quest for an efficient parallel implementation we must maximize the proportion of time the processors spend performing necessary computation. Any imbalance may result in processors standing idle while others struggle to complete their allocated work, thus limiting potential performance. Load balancing techniques aim to provide an even division of computational effort to all processors.

The solution of a problem using the domain decomposition model involves each processing element applying the specified algorithm to a set of principal data items. The computational model ensures that every principal data item is acted upon and determines how the tasks are allocated amongst the processing elements. A choice of computation model exists for each problem. To achieve maximum system performance, the model chosen must see that the total work load is distributed evenly amongst the processing elements. This balances the overheads associated with communicating principal data items to processing elements with the need to avoid processing element idle time. A simplified ray tracing example illustrate the differences between the computational models.

A sequential solution to this problem may be achieved by dividing the image plane into twenty-four distinct regions, with each region constituting a single principal data item, as shown in Figure 25, and then applying the ray tracing algorithm at each of these regions in turn. There are thus twenty-four tasks to be performed for this problem where each task is to compute the pixel value at one area of the image plane. To understand the computational models, it is not necessary to know the details of the algorithm suffice to say that each principal data item represents an area of the image plane on which the algorithm can be applied to determine the value for that position. We will assume that no additional data items are required to complete any task.

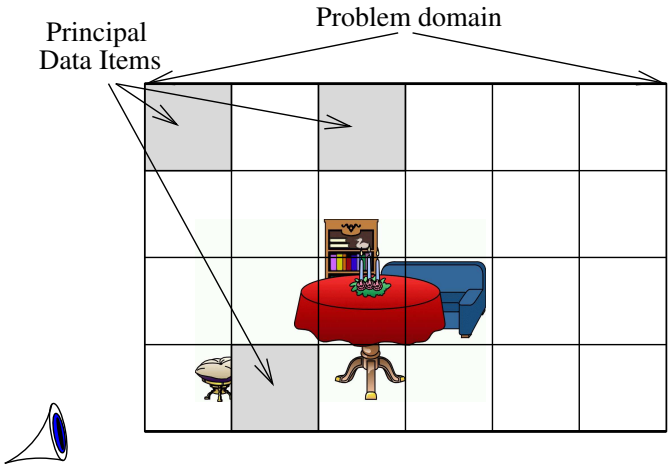


Figure 25: Principal data items for calculating the pixels in the image plane

8.4 Data Driven Model

The data driven model allocates all the principal data items to specific processing elements before computation commences. Each processing element thus knows *a priori* the principal data items to which they are required to apply the algorithm. Providing there is sufficient memory to hold the allocated set at each processing element, then, apart from the initial distribution, there is no further communication of principal data items. If there is insufficient local memory, then the extra items must be *fetched* as soon as memory space allows.

8.4.1 Balanced Data Driven

In balanced data driven systems (also known as geometric decompositions), an equal number of principal data items is allocated to each processing element. This portion is determined simply by dividing the total number of principal data items by the number of processing elements:

$$\text{portion at each PE} = \frac{\text{number of principal data items}}{\text{number of PEs}}$$

If the number of principal data items is not an exact multiple of the number of processing elements, then

$$(number\ of\ principal\ data\ items)\ MOD\ (number\ of\ PEs)$$

will each have one extra principal data item, and thus perform one extra task. The required start task and the number of tasks is communicated by the system controller to each processing element and these can then apply the required algorithm to their allotted principal data items. This is similar to the way in which problems are solved on arrays of SIMD processors.

In this example, consider the simple ray tracing calculation for an empty scene. The principal data items (the pixels) may be allocated equally to three processing elements, labeled PE_1, PE_2 and PE_3, as shown in Figure 26. In this case, each processing element is allotted eight principal data items.

As no further principal data item allocation takes place after the initial distribution, a balanced work load is only achieved for the balanced data driven computational model if the computational effort associated with each portion of principal data items is *identical*. If not, some processing elements will have finished their portions while others still have work to do. With the balanced data driven model the division of principal data items amongst processing elements is geometric in nature, that is each processing element simply may be allocated an equal number of principal data items irrespective of their position within the problem domain. Thus, to ensure a balanced work load, this model should only be used if the computational effort associated with each principal data item is the same, and preferably where the number of principal data items is an exact multiple of the number of processing elements. This implies *a priori* knowledge, but given this, the balanced data driven approach is the simplest of the computational models to implement.

Using Figure 26, if the computation of each pixel 1 *time unit* to complete, then the sequential solution of this problem would take 24 *time units*. The parallel implementation of this problem using the three processing elements each allocated eight tasks should take approximately 8 *time units*, a third of the time required by the sequential implementation. Note, however, that the parallel solution will not be exactly one third of the sequential time as this would ignore the time required to communicate the portions from the system controller to the processing elements. This also ignores time required to receive the results back from the processing elements and for the system controller to collate the solution. A balanced data driven version of this problem on the three processing elements would more accurately take:

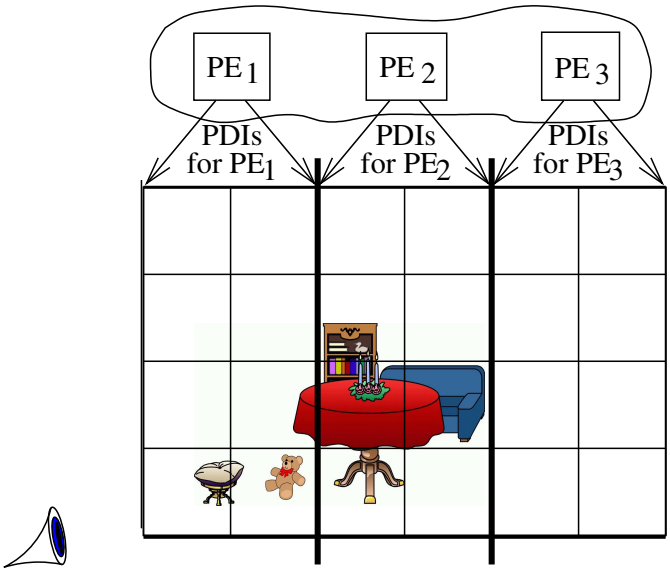


Figure 26: Equal allocation of data items to processing elements

$$\text{Solution time} = \text{initial distribution} + \lceil \frac{24}{3} \rceil + \text{result collation}$$

Assuming low communication times, this model gives the solution in approximately one third of the time of the sequential solution, close to the maximum possible linear speed-up. Solution of the same problem on five processing elements would give:

$$\text{Solution time} = \text{initial distribution} + \lceil \frac{24}{5} \rceil + \text{result collation}$$

This will be solved in even longer than the expected 4.8 *time units* as, in this case, one processing element is allocated 4 principal data items while the other four have to be apportioned 5. As computation draws to a close, one processing element will be idle while the four others complete their extra work. The solution time will thus be slightly more than 5 *time units*.

8.4.2 Unbalanced Data Driven

Differences in the computational effort associated with the principal data items will increase the probability of substantial processing element idle time if the simplistic balanced data driven approach is adopted. If the individual computation efforts differ, and are known *a priori*, then this can be exploited to achieve optimum load balancing.

The unbalanced data driven computational model allocates principal data items to processing elements based on their computational requirements. Rather than simply apportioning an equal number of tasks to each processing element, the principal data items are allocated to ensure that each processing element will complete its portion at approximately *the same time*.

For example, the complexity introduced into the ray tracing calculations by placing object into the scene, as shown in Figure 27, will cause an increased computational effort required to solve the portions allocated to PE_1 and PE_2 in the balanced data driven model. This will result in these two processing elements still being busy with their computations long after the other processing element, PE_3, has completed its less computationally complex portion.

Should *a priori* knowledge be available regarding the computational effort associated with each principal data item then they may be allocated *unequally* amongst the processing elements, as shown in Figure 28. The computational effort now required to process each of these unequal portions will be approximately the same, minimizing any processing element idle time.

The sequential time required to solve the ray tracing with objects in the scene is now 42 *time units*. To balance the work load amongst the three processing elements, each processing element should compute for 14 *time units*. Allocation of the portions to each processing element in the unbalanced data driven model involves a preprocessing step to determine precisely the best way to subdivide the principal data items. The optimum compute time for each processing element can be obtained by simply dividing the total computation time by the number of processing elements. If possible, no processing element should be allocated principal data items whose combined computation time exceeds this optimum amount. Sorting the principal data items in descending computation times can facilitate the subdivision.

The total solution time for a problem using the unbalanced data driven model is thus:

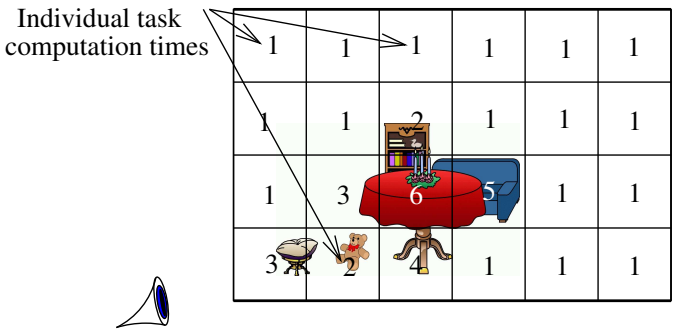


Figure 27: Unequal computational effort due to presence of objects in the scene

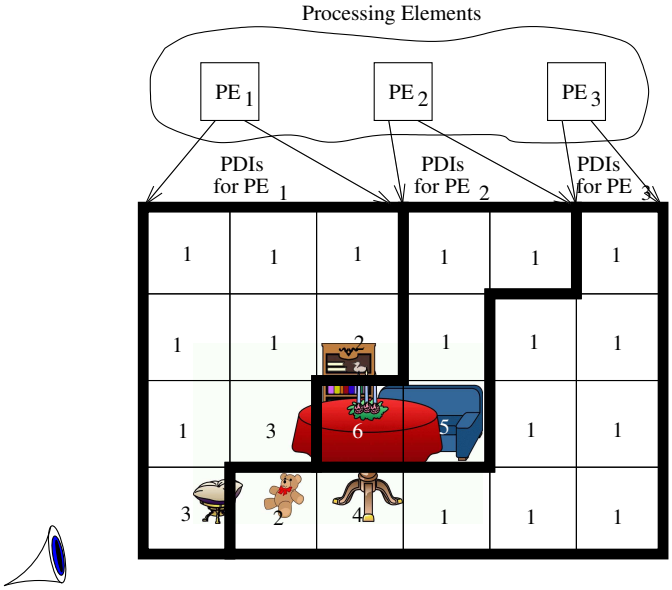


Figure 28: Unequal allocation of data items to processing elements to assist with load balancing

$$\begin{aligned}\text{Solution time} &= \text{preprocessing} + \text{distribution} \\ &\quad + \text{longest portion time} + \text{result collation}\end{aligned}$$

So comparing the naive balanced distribution from section 8.4

$$\begin{aligned}\text{Balanced solution time} &= \text{distribution} + 21 + \\ &\quad \text{result collation}\end{aligned}$$

$$\begin{aligned}&\text{Unbalanced solution time} = \\ &\quad \text{preprocessing} + \text{distribution} + 14 + \text{result collation}\end{aligned}$$

The preprocessing stage is a simple sort requiring far less time than the ray tracing calculations. Thus, in this example, the unbalanced data driven model would be significantly faster than the balanced model due to the large variations in task computational complexity.

The necessity for the preprocessing stage means that this model will take more time to use than the balanced data driven approach should the tasks have the same computation requirement. However, if there are variations in computational complexity and they are known, then the unbalanced data driven model is the most efficient way of implementing the problem in parallel.

8.4.3 Demand Driven Model

The data driven computational models are dependent on the computational requirements of the principal data items being known, or at least being predictable, before actual computation starts. Only with this knowledge can these data items be allocated in the correct manner to ensure an even load balance. Should the computational effort of the principal data items be unknown or unpredictable, then serious load balancing problems can occur if the data driven models are used. In this situation the demand driven computational model should be adopted to allocate work to processing elements evenly and thus optimize system performance.

In the demand driven computational model, work is allocated to processing elements *dynamically* as they become idle, with processing elements no longer bound to any particular portion of the principal data items. Having produced the result from one principal data item, the processing elements demand the next principal data item from some work supplier process. This is shown diagrammatically in Figure 29 for the simple ray tracing calculation.

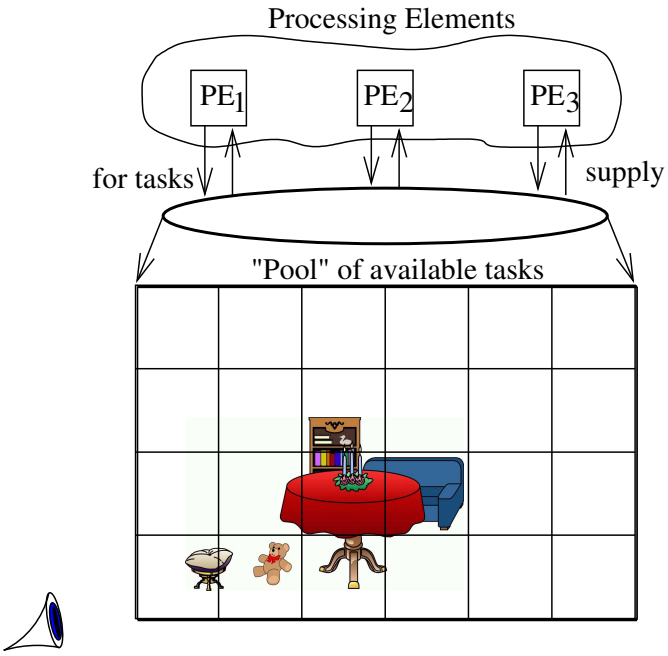


Figure 29: A demand driven model for a simple ray tracing calculation

Unlike the data driven models, there is no initial communication of work to the processing elements, however, there is now the need to send requests for individual principal data items to the supplier and for the supplier to communicate with the processing elements in order to satisfy these requests. To avoid unnecessary communication it may be possible to combine the return of the results from one computation with the request for the next principal data item.

The optimum time for solving a problem using this simple demand driven model is thus:

$$\text{Solution time} = 2 \times \text{total communication time} + \frac{\text{total computation time for all PDIs}}{\text{number of PEs}}$$

This optimum computation time, $\frac{\text{total computation time for all PDIs}}{\text{number of PEs}}$, will only be possible if the work can be allocated so that all processing elements complete the last of their tasks at exactly the same time. If this is not so then some processing elements will still be busy with their final task while the others have completed. It may also be possible to reduce the communication overheads of the demand driven model by overlapping the communication with the computation in some manner. This possibility will be discussed later in section 8.5.

On receipt of a request, if there is still work to be done, the work supplier responds with the next available task for processing. If there are no more tasks which need to be computed then the work supplier may safely ignore the request. The problem will be solved when all principal data items have been requested and all the results of the computations on these items have been returned and collated. The dynamic allocation of work by the demand driven model will ensure that while some processing elements are busy with more computationally demanding principal data items, other processing elements are available to compute the less complex parts of the problem.

Using the computational times for the presence of objects in the scene as shown in Figure 28, Figure 30 shows how the principal data items may be allocated by the task supplier to the processing elements using a simple serial allocation scheme. Note that the processing elements

do not complete the same number of tasks. So, for example, while processing elements 2 and 3 are busy completing the computationally complex work associated with principal data items 15 and 16, processing elements 1 can compute the less computationally taxing tasks of principal data items 17 and 18.

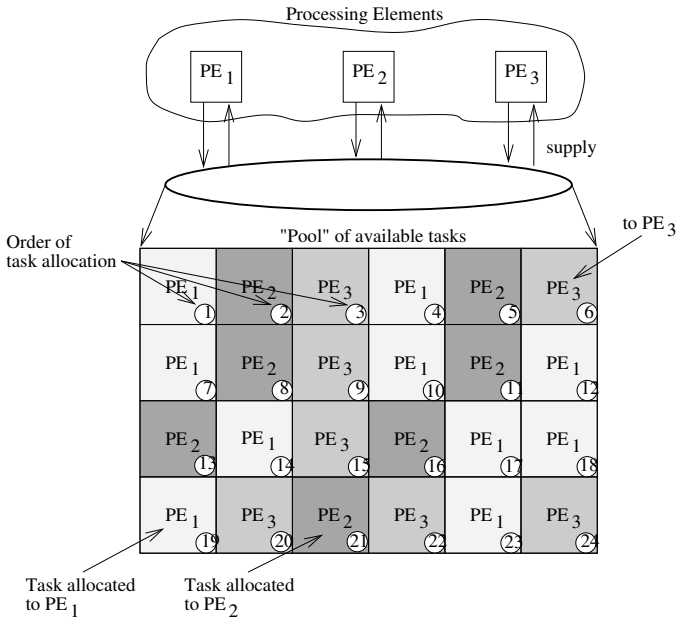


Figure 30: Allocation of principal data items using a demand driven model

The demand driven computational model facilitates dynamic load balancing when there is no prior knowledge as to the complexity of the different parts of the problem domain. Optimum load balancing is still dependent on all the processing elements completing the last of the work at the same time. An unbalanced solution may still result if a processing element is allocated a complex part of the domain towards the end of the solution. This processing element may then still be busy well after the other processing elements have completed computation on the remainder of the principal data items and are now idle as there is no further work to do. To reduce the likelihood of this situation it is important that the computationally complex portions of the domain, the so called *hot spots*, are allocated to processing elements early on in the solution process. Although there is no *a priori* knowledge as to the exact computational effort associated with any principal data item (if there were, an unbalanced data driven approach would have been adopted), nevertheless, any insight as to possible hot spot areas should be exploited. The task supplier would thus assign principal data items from these areas first.

In the ray tracing example , while the exact computational requirement associated with the principal data items in proximity of the objects in the scene may be unknown, it is highly likely that the solution of the principal items in that area will more complex than those elsewhere. In this problem, these principal data items should be allocated first.

If no insight is possible then a simple serial allocation, as shown in Figure 30, or spiral allocation or even a random allocation of principal data items will have to suffice. While a random allocation offers perhaps a higher probability of avoiding late allocation of principal data items from hot spots, additional effort is required when choosing the next principal data item to allocate to ensure that no principal data item is allocated more than once.

As with all aspects of parallel processing, extra levels of sophistication can be added in order to exploit any information that becomes available as the parallel solution proceeds. Identifying possible hot spots in the problem domain may be possible from the computation time associated with each principal data item as these become known. If this time is returned along with the result for that principal data item, the work supplier can build a dynamic profile of the computational requirements associated with areas of the domain. This information can be used to adapt the allocation scheme to send principal data items from the possible hot spot regions. There is, of course, a trade off here between the possible benefits to load balancing in the early allocation of principal data items from hot spots, and the overhead that is introduced by the need to:

- time each computation at the processing elements;
- return this time to the work supplier;
- develop the time profile at the work supplier; and,
- adapt the allocation strategy to take into account this profile.

The benefits gained by such an adaptive scheme are difficult to predict as they are dependent on the problem being considered and the efficiency of the scheme implementation. The advice in these matters is always: "*implement a simple scheme initially and then add extra sophistication should resultant low system performance justify it.*"

8.4.4 Hybrid Computational Model

For most problems, the correct choice of computational model will either be one of the data driven strategies or the demand driven approach. However, for a number of problems, a hybrid computational model, exhibiting properties of both data and demand driven models, can be adopted to achieve improved efficiency. The class of problem that can benefit from the hybrid model is one in which an initial set of principal data items of known computational complexity may spawn an unknown quantity of further work.

In this case, the total number of principal data items required to solve the problem is *unknown* at the start of the computation, however, there are at least a known number of principal data items that must be processed first. If the computational complexity associated with these initial principal data items is unknown then a demand driven model will suffice for the whole problem, but if the computational complexity is known then one of the data driven models, with their lower communication overheads, should at least be used for these initial principal data items. Use of the hybrid model thus requires the computational model to be switched from data driven to demand driven mode as required.

8.5 Task Management

Task management encompasses the following functions:

- the definition of a task;
- controlling the allocation of tasks;
- distribution of the tasks to the processing elements; and,
- collation of the results, especially in the case of a problem with multiple stages.

8.5.1 Task Definition and Granularity

An *atomic element* may be thought of as a problem's lowest computational element within the sequential algorithm adopted to solve the problem. As introduced in section 8.1.2, in the domain decomposition model a single task is the application of this sequential algorithm to a principal data item to produce a result for the sub-parts of the problem domain. The task is thus the smallest element of computation for the problem within the parallel system. The *task granularity* (or *grain size*) of a problem is the number of atomic elements, which are included in one task. Generally, the task granularity remains constant for all tasks, but in some cases it may be desirable to alter dynamically this granularity as the computation proceeds. A task which includes only one atomic element is said to have the *finest granularity*, while a task which contains many is *coarser grained*, or has a *coarser granularity*. The actual definition of what constitutes a principal data item is determined by the granularity of the tasks.

A parallel system solves a problem by its constituent processing elements executing tasks in parallel. A *task packet* is used to inform a processing element which task, or tasks, to perform. This task packet may simply indicate which tasks require processing by that processing element, thus forming the lowest level of distributed work. The packet may include additional information, such as additional data items, which the tasks require in order to be completed.

To illustrate the differences in this terminology, consider again the simple ray tracing problem. The atomic element of a sequential solution of this problem could be to perform a single ray-object intersection test. The principal data item is the pixel being computed and the additional data item required will be object being considered. A sequential solution of this problem would be for a single processing element to consider each ray-object intersection in turn. The help of several processing elements could substantially improve the time taken to perform the ray tracing.

The finest task granularity for the parallel implementation of this problem is for each task to complete one atomic element, that is perform one ray-object intersection. For practical considerations, it is perhaps more appropriate that each task should instead be to trace the complete path of a single ray. The granularity of each task is now the number of ray-object intersections required to trace this single ray and each pixel is a principal data item. A sensible task packet to distribute the work to the processing elements would include details about one or more pixels together with the necessary scene data.

To summaries our choices for this problem:

atomic element: to perform one ray-object intersection;

task: to trace the complete path of one ray (may consists of a number of atomic elements);

PDI: the pixel location for which we are computing the color;

ADI: the scene data; and,

task packet: one or more rays to be computed.

Choosing the task granularity for the parallel implementation of a problem is not straightforward. Although it may be fairly easy to identify the atomic element for the sequential version of the problem, such a fine grain may not be appropriate when using many processing elements. Although the atomic element for ray tracing was specified as computing a single ray-object intersection in the above example, the task granularity for the parallel solution was chosen as computing the complete color contribution at a particular pixel. If one atomic element had been used as the task granularity then additional problems would have introduced for the parallel solution, namely, the need for processors

to to exchange partial results. This difficulty would have been exacerbated if, instead, the atomic element had been chosen as tracing a ray into a voxel and considering whether it does in fact intersect with an object there. Indeed, apart from the higher communication overhead this would have introduced, the issue of dependencies would also have to be checked to ensure, for example, that a ray was not checked against an object more than once.

As well as introducing additional communication and dependency overheads, the incorrect choice of granularity may also increase computational complexity variations and hinder efficient load balancing. The choice of granularity is seldom easy, however, a number of parameters of the parallel system can provide an indication as to the desirable granularity. The computation to communication ratio of the architecture will suggest whether additional communication is acceptable to avoid dependency or load balancing problems. As a general rule, where possible, data dependencies should be avoided in the choice of granularity as these imply unnecessary synchronization points within the parallel solution which can have a significant effect on overall system performance.

8.5.2 Task Distribution and Control

The task management strategy controls the distribution of packets throughout the system. Upon receipt, a processing element performs the tasks specified by a packet. The composition of the task packet is thus an important issue that must be decided before distribution of the tasks can begin. To complete a task a processing element needs a copy of the algorithm, the principal data item(s), and any additional data items that the algorithm may require for that principal data item. The domain decomposition paradigm provides each processing element with a copy of the algorithm, and so the responsibility of the task packet is to provide the other information.

The principal data items form part of the problem domain. If there is sufficient memory, it may be possible to store the entire problem domain as well as the algorithm at each processing element. In this case, the inclusion of the principal data item as part of the task packet is unnecessary. A better method would be simply to include the identification of the principal data item within the task packet. Typically, the identification of a principal data item is considerably smaller, in terms of actual storage capacity, than the item itself. The communication overheads associated with sending this smaller packet will be significantly less than sending the principal data item with the packet. On receipt of the packet the processing element could use the identification simply to fetch the principal data item from its local storage. The identification of the principal data item is, of course, also essential to enable the results of the entire parallel computation to be collated.

If the additional data items required by the task are known then they, or if possible, their identities, may also be included in the task packet. In this case the task packet would form an integral unit of computation which could be directly handled by a processing element. However, in reality, it may not be possible to store the whole problem domain at every processing element. Similarly, numerous additional data items may be required which would make their inclusion in the task packet impossible. Furthermore, for a large number of problems, the additional data items which are required for a particular principal data item may not be known in advance and will only become apparent as the computation proceeds.

A task packet should contain as a minimum either the identity, or the identity and actual principal data items of the task. The inability to include the other required information in the packet means that the parallel system will have to resort to some form of *data management*.

8.5.3 Algorithmic Dependencies

The algorithm of the problem may specify an order in which the work must be undertaken. This implies that certain tasks must be completed before others can commence. These dependencies must be preserved in the parallel implementation. In the worst case, algorithmic dependencies can prevent an efficient parallel implementation, as shown with the tower of toy blocks in Figure 11. Amdahl's law, described in section 7.4, shows the implications to the algorithmic decomposition model of parallel processing of the presence of even a small percentage of purely sequential code. In the domain decomposition approach, algorithmic dependencies may introduce two phenomena which will have to be tackled:

- *synchronization points* which have the effect of dividing the parallel implementation into a number of distinct stages; and,
- *data dependencies* which will require careful data management to ensure a consistent view of the data to all processing elements.

Multi-Stage Algorithms

Many problems can be solved by a single stage of computation, utilizing known principal data items to produce the desired results. However, the dependencies inherent in other algorithms may divide computation into a number of distinct stages. The *partial results* produced by one stage become the principal data items for the following stage of the algorithm, as shown in Figure 31. For example, many scientific problems involve the construction of a set of simultaneous equations, a distinct stage, and the subsequent solution of these equations for the unknowns. The partial results, in this case elements of the simultaneous equations, become the principal data for the tasks of the next stage.

Even a single stage of a problem may contain a number of distinct sub-stages which must first be completed before the next sub-stage can proceed. An example of this is the use of an iterative solver, such as the Jacobi method, to solve a set of simultaneous equations. An iterative method starts with an approximate solution and uses it in a recurrence formula to provide another approximate solution. By repeatedly applying this process a sequence of solutions is obtained which, under suitable conditions, converges towards the exact solution.

Consider the problem of solving a set of six equations for six unknowns, $\mathbf{A}x = \mathbf{b}$. The Jacobi method will solve this set of equations by calculating, at each iteration, a new approximation from the values of the previous iteration. So the value for the x_i's at the n^{th} iteration are calculated as:

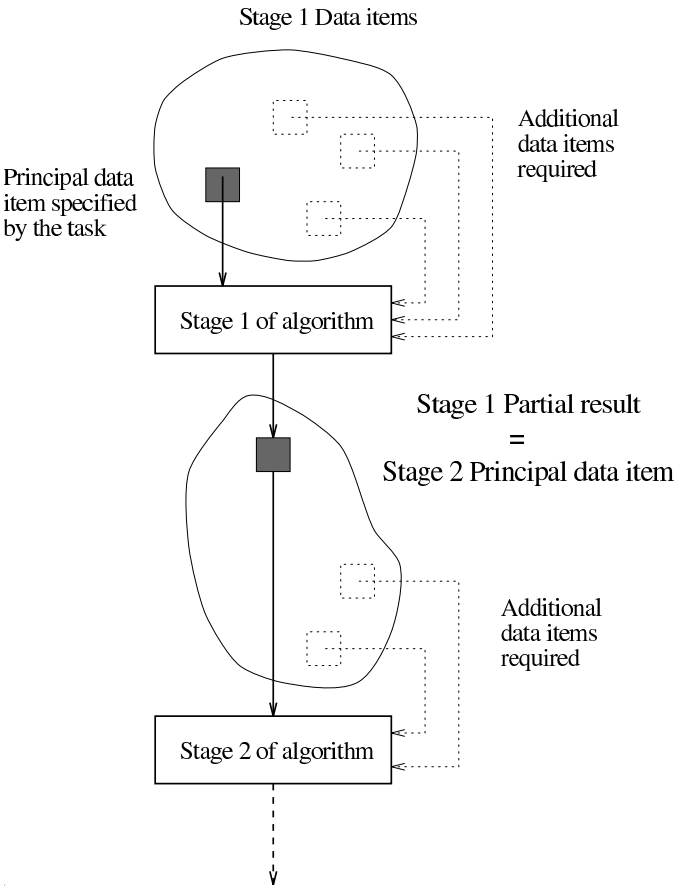


Figure 31: The introduction of partial results due to algorithmic dependencies

$$
\begin{aligned}
x_1^n &= \frac{b_i - a_{12}x_2^{n-1} - \ldots - a_{16}x_6^{n-1}}{a_{11}} \\
x_2^n &= \frac{b_i - a_{21}x_1^{n-1} - \ldots - a_{26}x_6^{n-1}}{a_{22}} \\
&\vdots \\
x_6^n &= \frac{b_i - a_{61}x_1^{n-1} - \ldots - a_{65}x_6^{n-1}}{a_{66}}
\end{aligned}
$$

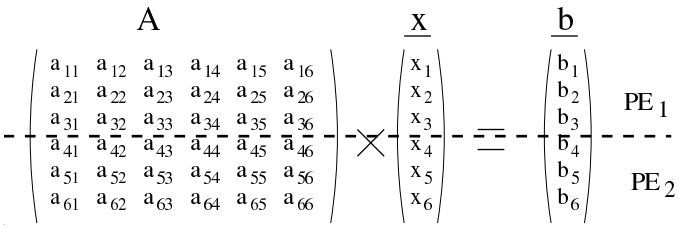


Figure 32: Solving an iterative matrix solution method on two processing elements

A parallel solution to this problem on two processing elements could allocate three rows to be solved to each processing element as shown in Figure 32. Now PE_1 can solve the n^{th} iteration values x_1^n, x_2^n and x_3^n in parallel with PE_2 computing the values of x_4^n, x_5^n and x_6^n. However, neither processing element can proceed onto the $(n+1)^{st}$ iteration until both have finished the n^{th} iteration and exchanged their new approximations for the x_i^n's. Each iteration is, therefore, a sub-stage which must be completed before the next sub-stage can commence. This point is illustrated by the following code segment from PE_1:

```
PROCEDURE Jacobi() (* Executing on PE 1 *)
    Begin
        Estimate x[1] ... x[6]
        n := 0 (* Iteration number *)
        WHILE solution_not_converged DO
            Begin
                n := n + 1
                Calculate new x[1], x[2] & x[3] using old x[1] ... x[6]
                PARALLEL
                    SEND new x[1], x[2] & x[3] TO PE_2
                    RECEIVE new x[4], x[5] & x[6] FROM PE_2
            End
    End (* Jacobi *)
```

Data Dependencies

The concept of dependencies was introduced in section 6.1 when we were unable to construct a tower of blocks in parallel as this required a strictly sequential order of task completion. In the domain decomposition model, data dependencies exist when a task may not be performed on some principal data item until another task has been completed. There is thus an implicit ordering on the way in which the task packets may be allocated to the processing elements. This ordering will prevent certain tasks being allocated, even if there are processing elements idle, until the tasks on which they are dependent have completed.

A linear dependency exists between each of the iterations of the Jacobi method discussed above. However, no dependency exists for the calculation of each x_i^n, for all i, as all the values they require, x_j^{n-1}, $\forall j \neq i$, will already have been exchanged and thus be available at every processing element.

The Gauss-Seidel iterative method has long be preferred in the sequential computing community as an alternative to Jacobi. The Gauss-Seidel method makes use of new approximations for the x_i as soon as they are available rather than waiting for the next iteration. Provided the methods converge, Gauss-Seidel will converge more rapidly than the Jacobi method. So, in the example of six unknowns above, in the n^{th} the value of x_1^n would still be calculated as:

$$x_1^n = \frac{b_i - a_{12}x_2^{n-1} - \ldots - a_{16}x_6^{n-1}}{a_{11}},$$

but the x_2^n value would now be calculated by:

$$x_2^n = \frac{b_i - a_{21}x_1^n - a_{23}x_3^{n-1} - \ldots - a_{26}x_6^{n-1}}{a_{22}}$$

Although well suited to sequential programming, the strong linear dependency that has been introduced, makes the Gauss-Seidel method poorly suited for parallel implementation. Now within each iteration no value of x_i^n can be calculated until all the values for x_j^n, $j < i$ are available; a strict sequential ordering of the tasks. The less severe data dependencies within the Jacobi method thus make it a more suitable candidate for parallel processing than the Gauss-Seidel method which is more efficient on a sequential machine.

It is possible to implement a hybrid of these two methods in parallel, the so-called "Block Gauss-Seidel - Global Jacobi" method. A processing element which is computing several rows of the equations, may use the Gauss-Seidel method for these rows as they will be computed sequentially within the processing element. Any values for x_i^n not computed locally will assume the values of the previous iteration, as in the Jacobi method. All new approximations will be exchanged at each iteration. So, in the example, PE_2 would calculate the values of x_4^n, x_5^n and x_6^n as follows:

$$x_4^n = \frac{b_i - a_{11}x_1^{n-1} - a_{12}x_2^{n-1} - a_{13}x_3^{n-1} - a_{15}x_5^{n-1} - a_{16}x_6^{n-1}}{a_{44}}$$

$$x_5^n = \frac{b_i - a_{11}x_1^{n-1} - a_{12}x_2^{n-1} - a_{13}x_3^{n-1} - a_{14}x_4^{n} - a_{16}x_6^{n-1}}{a_{55}}$$

$$x_6^n = \frac{b_i - a_{11}x_1^{n-1} - a_{12}x_2^{n-1} - a_{13}x_3^{n-1} - a_{14}x_4^{n} - a_{15}x_5^{n}}{a_{66}}$$

8.6 Task Scheduling Strategies

8.6.1 Data Driven Task Management Strategies

In a data driven approach, the system controller determines the allocation of tasks prior to computation proceeding. With the unbalanced strategy, this may entail an initial sorting stage based on the known computational complexity, as described in section 8.4.2. A single task-packet detailing the tasks to be performed is sent to each processing element. The application processes may return the results upon completion of their allocated portion, or return individual results as each task is performed, as shown in this code segment:

```
PROCESS Application_Process()
    Begin
      RECEIVE task_packet FROM SC via R
      FOR i = start_task_id TO finish_task_id DO
        Begin
          result[i] := Perform_Algorithm(task[i])
          SEND result[i] TO SC via R
        End
    End (* Application_Process *)
```

In a data driven model of computation a processing element may initially be supplied with as many of its allocated principal data items as its local memory will allow. Should there be insufficient storage capacity a simple data management strategy may be necessary to prefetch the missing principal data items as computation proceeds and local storage allows.

8.6.2 Demand Driven Task Management Strategies

Task management within the demand driven computational model is explicit. The work supplier process, which forms part of the system controller, is responsible for placing the tasks into packets and sending these packets to requesting processing elements. To facilitate this process, the system controller maintains a *pool* of already constituted task packets. On receipt of a request, the work supplier simply dispatches the next available task packet from this task pool, as can be seen in Figure 33.

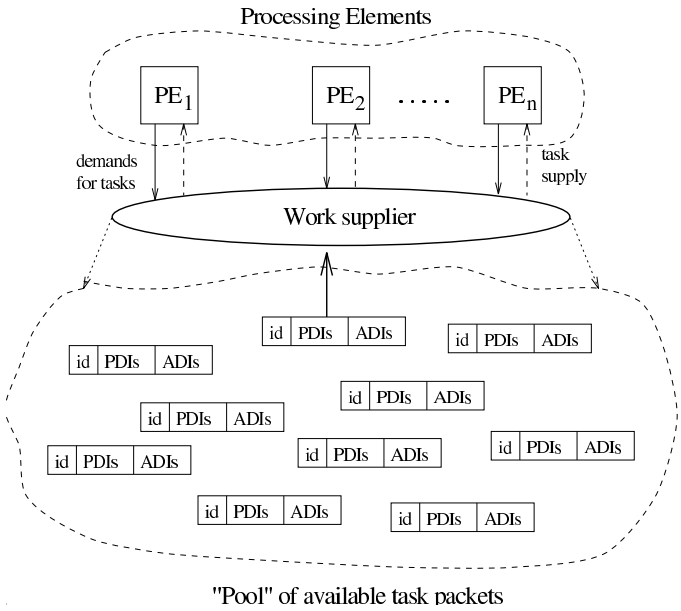


Figure 33: Supplying task packets from a task pool at the system controller

The advantage of a task pool is that the packets can be inserted into it in advance, or concurrently as the solution proceeds, according to the allocation strategy adopted. This is especially useful for problems that create work dynamically, such as those using the hybrid approach as described in section 8.4.4. Another advantage of the task pool is that if a hot spot in the problem domain is identified, then the ordering within the task pool can be changed dynamically to reflect this and thus ensure that potentially computationally complex tasks are allocated first.

More than one task pool may be used to reflect different levels of task priority. High priority tasks contained in the appropriate task pool will always be sent to a requesting processing element first. Only once this high priority task pool is (temporarily) empty will tasks from lower priority pools be sent. The multiple pool approach ensures that high priority tasks are not ignored as other tasks are allocated.

In the demand driven computational model, the processing elements demand the next task as soon as they have completed their current task. This demand is translated into sending a request to the work supplier, and the demand is only satisfied when the work supplier has delivered

the next task. There is thus a definite delay period from the time the request is issued until the next task is received. During this period the processing element will be computationally idle. To avoid this idle time, it may be useful to include a buffer at each processing element capable of holding at least one task packet. This buffer may be considered as the processing element's own private task pool. Now, rather than waiting for a request to be satisfied from the remote system controller, the processing element may proceed with the computation on the task packet already present locally. When the remote request has been satisfied and a new task packet delivered, this can be stored in the buffer waiting for the processing element to complete the current task.

Whilst avoiding delays in fetching tasks from a remote task pool, the use of a buffer at each processing element may have serious implications for load balancing, especially towards the end of the problem solution. We will examine these issues in more detail after we have considered the realization of task management for a simple demand driven system - the processor farm.

A First Approach: The Processor Farm

Simple demand driven models of computation have been implemented and used for a wide range of applications. One realization of such a model, often referred to in the literature, is that implemented by May and Shepherd [MS87]. This simple demand driven model, which they term a *processor farm*, has been used for solving problems with high computation to communication ratios. The model proposes a single system controller and one or more processing elements connected in a linear configuration, or chain. The structure of a processing element in this model is shown in Figure 34.

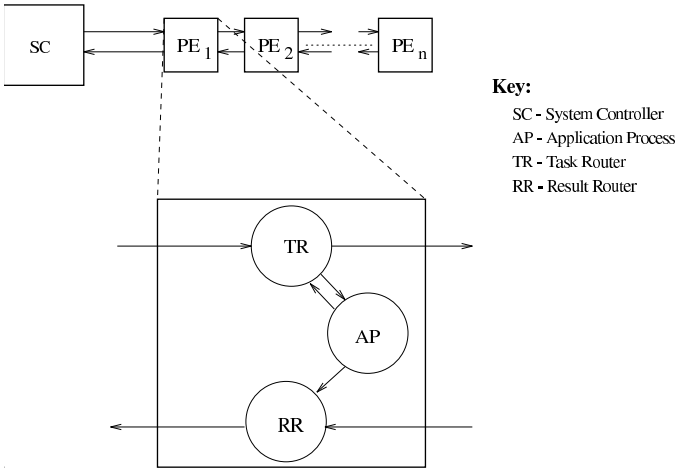


Figure 34: A processing element for the processor farm model

The application process performs the desired computation, while the communication within the system is dealt with by two router processes, the Task Router (TR) and the Result Router (RR). As their names suggest, the task router is responsible for distributing the tasks to the application process, while the result router returns the results from the completed tasks back to the system controller. The system controller contains the initial pool of tasks to be performed and collates the results. Such a communication strategy is simple to implement and largely problem independent.

To reduce possible processing element idle time, each task router process contains a single buffer in which to store a task so that a new task can be passed to the application process as soon as it becomes idle. When a task has been completed the results are sent to the system controller. On receipt of a result, the system controller releases a new task into the system. This synchronized releasing of tasks ensures that there are never more tasks in the system than there is space available.

On receipt of a new task, the task router process either:

1. passes the task directly to the application process if it is waiting for a task; or
2. places the task into its buffer if the buffer is empty; or, otherwise
3. passes the task onto the next processing element in the chain.

The processor farm is initialized by loading sufficient tasks into the system so that the buffer at each task router is full and each application process has a task with which to commence processing. Figure 35 shows the manner in which task requests are satisfied within a simple two processing element configured in a chain.

The simplicity of this realization of a demand driven model has contributed largely to its popularity. Note that because of the balance maintained within the system, the only instance at which the last processing element is different from any other processing element in the chain is to ensure the `closedown_command` does not get passed any further. However, such a model does have disadvantages which may limit its use for more complex problems.

The computation to communication ratio of the desired application is critical in order to ensure an adequate performance of a processor farm. If this ratio is too low then significant processing element idle time will occur. This idle time occurs because the computation time for the

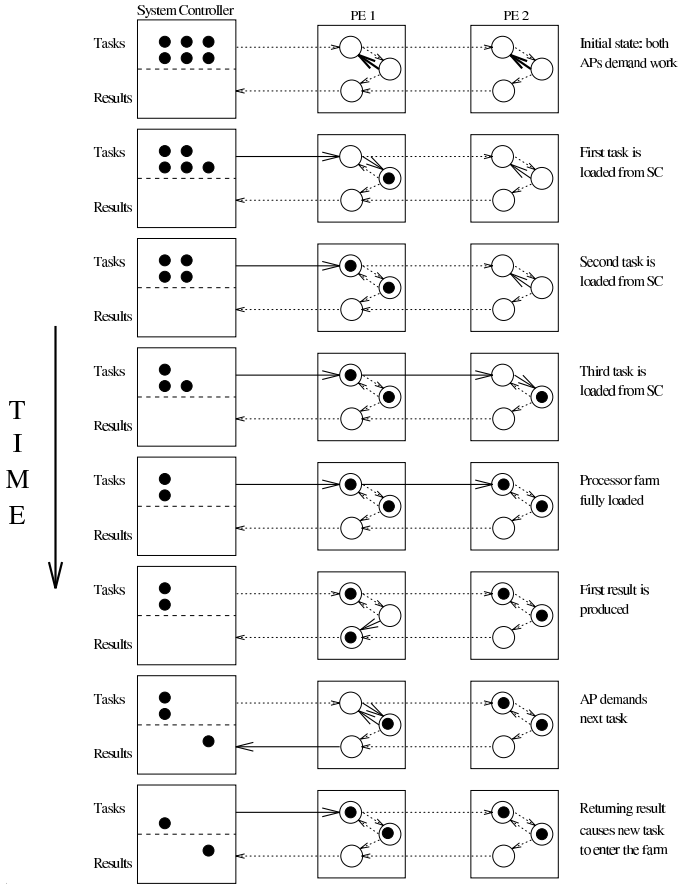


Figure 35: Task movement within a two PE processor farm

application process to complete its current task and the task buffered at the task router may be lower than the combined communication time required for the results to reach the system controller plus the time for the new tasks released into the system to reach the processing element. This problem may be partially alleviated by the inclusion of several buffers at each task router instead of just one. However, without *a priori* knowledge as to the computation to communication ratio of the application, it may be impossible to determine precisely what the optimum number of buffers should be. This analysis is particularly difficult if the computational complexity of the tasks vary; precisely the type of problem demand driven models are more apt at solving. The problem independence of the system will also be compromised by the use of any *a priori* knowledge.

If the number of buffers chosen is too small, then the possibility of application process idle time will not be avoided. Provision of too many buffers will certainly remove any immediate application process idle time, but will re-introduce the predicament as the processing draws to a close. This occurs once the system controller has no further tasks to introduce into the system and now processing must only continue until all tasks still buffered at the processing elements have been completed. Obviously, significant idle time may occur as some processing elements struggle to complete their large number of buffered tasks.

The computation to communication ratio of the processor farm is severely exacerbated by the choice of the chain topology. The distance between the furthest processing element in the chain and the system controller grows linearly as more processing elements are added. This means that the combined communication time to return a result and receive a new task also increases. Furthermore, this communication time will also be adversely affect by the message traffic of all the intermediate processing elements which are closer to the system controller.

8.7 Task Manager Process

The aim of task management within a parallel system is to ensure the efficient supply of tasks to the processing elements. A Task Manager process (TM) is introduced at each processing element to assist in maintaining a continuous supply of tasks to the application process. The application process no longer deals with task requests directly, but rather indirectly using the facilities of the task manager. The task manager process assumes the responsibility for ensuring that every request for additional tasks from the application process will be satisfied immediately. The task manager attempts to achieve this by maintaining a local task pool.

In the processor farm, the task router process contains a single buffered task in order to satisfy the next local task request. As long as this buffer is full, task supply is immediate as far as the application process is concerned. The buffer is refilled by a new task from the system

controller triggered on receipt of a result. The task router acts in a *passive* manner, awaiting replenishment by a new task within the farm. However, if the buffer is empty when the application process requests a task then this process must remain idle until a new task arrives. This idle time is wasted computation time and so to improve system performance the passive task router should be replaced by a "intelligent" task manager process more capable of ensuring new tasks are always available locally.

The task management strategies implemented by the task manager and outlined in the following sections are *active*, dynamically requesting and acquiring tasks during computation. The task manager thus assumes the responsibility of ensuring local availability of tasks. This means that an application process should *always* have its request for a task satisfied immediately by the task manager unless:

- at the start of the problem the application processes make a request before the initial tasks have been provided by the system controller;
- there are no more tasks which need to be solved for a particular stage of the parallel implementation; or,
- the task manager's replenishment strategy has failed in some way.

8.7.1 A Local Task Pool

To avoid any processing element idle time, it is essential that the task manager has at least one task available locally at the moment the application process issues a task request. This desirable situation was achieved in the processor farm by the provision of a single buffer at each task router. As we saw, the single buffer approach is vulnerable to the computation to communication ratio within the system. Adding more buffers to the task router led to the possibility of serious load imbalances towards the end of the computation.

The task manager process maintains a local task pool of tasks awaiting computation by the application process. This pool is similar to the task pool at the system controller, as shown in Figure 33. However, not only will this local pool be much smaller than the system controller's task pool, but also it may be desirable to introduce some form of "status" to the number of available tasks at any point in time.

Satisfying a task request will free some space in the local task pool. A simple replenishment strategy would be for the task manager immediately to request a new task packet from the system controller. This request has obvious communication implications for the system. If the current message densities within the system are high and as long as there are still tasks available in the local task pool, this request will place and unnecessary additional burden on the already overloaded communication network.

As an active process, it is quite possible for the task manager to delay its replenishment request until message densities have diminished. However, this delay must not be so large that subsequent application process demands will deplete the local task pool before any new tasks can be fetched causing processor idle time to occur. There are a number of indicators which the task manager can use to determine a suitable delay. Firstly, this delay is only necessary if current message densities are high. Such information should be available for the router. Given a need for delay, the number of tasks in the task pool, the approximate computation time each of these tasks requires, and the probable communication latency in replenishing the tasks should all contribute to determining the request delay.

In a demand driven system, the computational complexity variations of the tasks are not known. However, the task manager will be aware of how long previous tasks have taken to compute (the time between application process requests). Assuming some form of preferred biased allocation of tasks in which tasks from similar regions of the problem domain are allocated to the same processing element, as discussed in section 8.9, the task manager will be able to build up a profile of task completion time which can be used to predict approximate completion times for tasks in the task pool. The times required to satisfy previous replenishment requests will provide the task manager with an idea of likely future communication responses. These values are, of course, mere approximations, but they can be used to assist in determining reasonable tolerance levels for the issuing of replenishment requests.

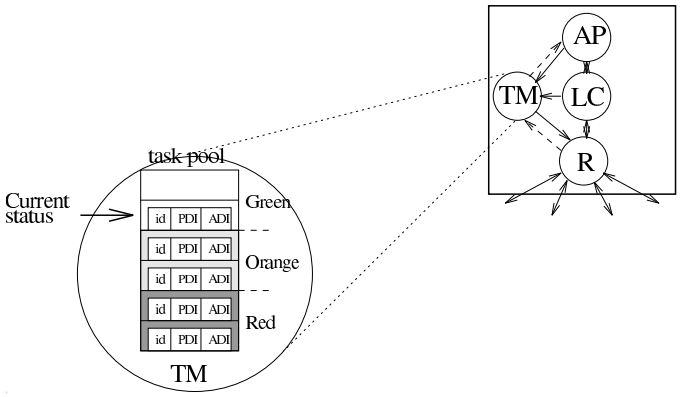


Figure 36: Status of task manager's task pool

The task manager's task pool is divided into three regions: *green*, *orange* and *red*. The number of tasks available in the pool will determine the current status level, as shown in Figure 36. When faced with the need to replenish the task pool the decision can be taken based on the current status of the pool:

green: Only issue the replenishment request if current message traffic density is low;

orange: Issue the replenishment request unless the message density is very high; and,

red: Always issue the replenishment request.

The boundaries of these regions may be altered dynamically as the task manager acquires more information. At the start of the computation the task pool will be all red. The computation to communication ratio is critical in determining the boundaries of the regions of the task pool. The better this ratio, that is when computation times are high relative to the time taken to replenish a task packet, the smaller the red region of the task pool need be. This will provide the task manager with greater flexibility and the opportunity to contribute to minimizing communication densities.

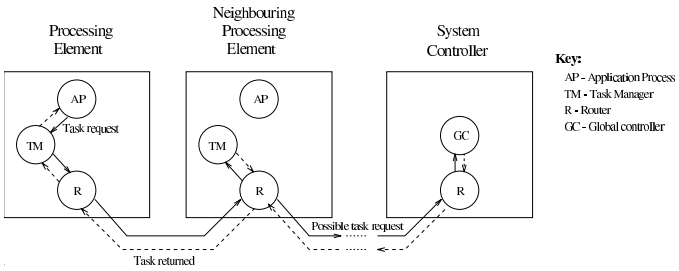


Figure 37: Task request propagating towards the system controller

8.8 Distributed Task Management

One handicap of the centralized task pool system is that all replenishment task requests from the task managers must reach the system controller before the new tasks can be allocated. The associated communication delay in satisfying these requests can be significant. The communication problems can be exacerbated by the bottleneck arising near the system controller. Distributed task management allows task requests to be handled at a number of locations remote from the system controller. Although all the tasks originate from the system controller, requests from processing elements no longer have to reach there in order to be satisfied.

The closest location for a task manager to replenish a task packet is from the task pool located at the task manager of one of its nearest neighbors. In this case, a replenishment request no longer proceeds directly to the system controller, but simply via the appropriate routers to the neighboring task manager. If this neighboring task manager is able to satisfy the replenishment request then it does so from its task pool. This task manager may now decide to in turn replenish its task pool, depending on its current status and so it will also request another task from one of its neighboring task managers, but obviously not the same neighbor to which it has just supplied the task. One sensible strategy is to propagate these requests in a “chain like” fashion in the direction towards the main task supplier at the system controller, as shown in Figure 37.

This distributed task management strategy is referred to as a *producer-consumer* model. The application process is the initial consumer and its local task manager the producer. If a replenishment request is issued then this task manager becomes the consumer and the neighboring task manager the producer, and so on. The task supplier process of the system controller is the overall producer for the system. If no further tasks exist at the system controller then the last requesting task manager may change the direction of the search. This situation may occur towards the end of a stage of processing and facilitates load balancing of any tasks remaining in task manager buffers. As well as reducing the communication distances for task replenishment, an additional advantage of this “chain reaction” strategy is that the number of request messages in the system is reduced. This will play a major rôle helping maintain a lower overall message density within the system.

If a task manager is unable to satisfy a replenishment request as its task pool is empty, then to avoid “starvation” at the requesting processing element, this task manager must ensure that the request is passed on to another processing element.

A number of variants of the producer-consumer model are also possible:

- Instead of following a path towards the system controller, the “chain reaction” could follow a predetermined Hamiltonian path (the system controller could be one of the processors on this path).

  Aside: A Hamiltonian path is a circuit starting and finishing at one processing element. This circuit passes through each processor in the network once only.

  Such a path would ensure that a processing element would be assured of replenishing a task if there was one available and there would be no need to keep track of the progress of the “chain reaction” to ensure no task manager was queried more than once per chain.

- In the course of its through-routing activities a router may handle a task packet destined for a distant task manager. If that router’s local task manager has an outstanding “red request” for a task then it is possible for the router to *poach* the “en route task” by diverting it, so satisfying its local task manager immediately. Care must be taken to ensure that the task manager for whom the task was intended is informed that the task has been poached, so it may issue another request. In general, tasks should only be poached from “red replenishment” if to do so would avoid local application process idle time.

8.9 Preferred Bias Task Allocation

The preferred bias method of task management is a way of allocating tasks to processing elements which combines the simplicity of the balanced data driven model with the flexibility of the demand driven approach. To reiterate the difference in these two computational models as they pertain to task management:

- Tasks are allocated to processing elements in a predetermined manner in the balanced data driven approach.
- In the demand driven model, tasks are allocated to processing elements on demand. The requesting processing element will be assigned the next available task packet from the task pool, and thus no processing element is bound to any area of the problem domain.

Provided no data dependencies exist, the order of task completion is unimportant. Once all tasks have been computed, the problem is solved. In the preferred bias method the problem domain is divided into equal regions with each region being assigned to a particular processing element, as is done in the balanced data driven approach. However, in this method, these regions are purely *conceptual* in nature. A demand driven model of computation is still used, but the tasks are not now allocated in an arbitrary fashion to the processing elements. Rather, a task is dispatched to a processing element from its conceptual portion. Once all tasks from a processing element's conceptual portion have been completed, only then will that processing element be allocated its next task from the portion of another processing element which has yet to complete its conceptual portion of tasks. Generally this task should be allocated from the portion of the processing element that has completed the least number of tasks. So, for example, from Figure 38, on completion of the tasks in its own conceptual region, PE_3 may get allocated task number 22 from PE_2's conceptual region. Preferred bias allocation is sometimes also referred to as *conceptual task allocation*.

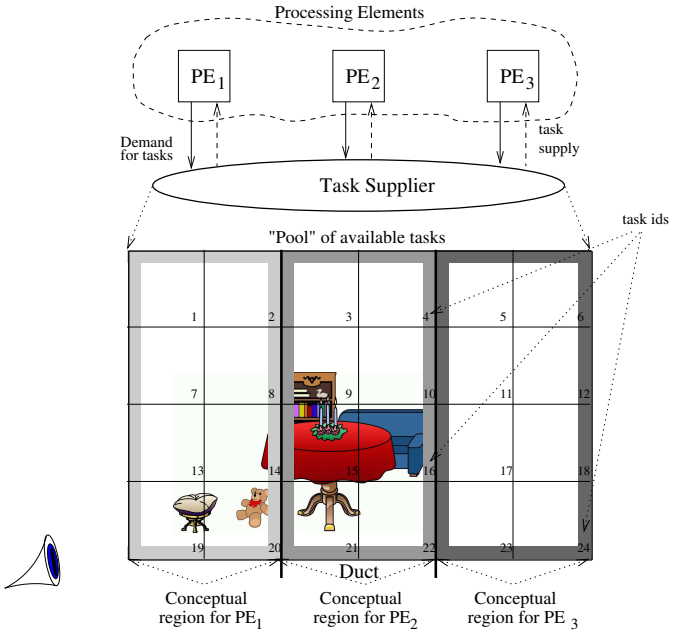


Figure 38: Partial result storage balancing by means of conceptual regions

The implications of preferred bias allocation are substantial. The demand driven model's ability to deal with variations in computational complexity is retained, but now the system controller and the processing elements themselves know to whom a task that they have been allocated conceptually belongs. This can greatly facilitate the even distribution of partial results at the end of any stage of a multi-stage problem.

The exploitation of data coherence is a vital ploy in reducing idle time due to remote data fetches. Preferred bias allocation of tasks can ensure that tasks from the same region of the problem are allocated to the same processing element. This can greatly improve the cache hit ratio at that processing element.

Part IV

Applications in Archeology

Computer reconstructions of heritage sites provide us with a means of visualizing past environments, allowing us a glimpse of the past that might otherwise be difficult to appreciate. However, it is essential that these reconstructions incorporate all the physical evidence for the site, otherwise there is a very real danger of misrepresenting the past [CD02]. A key feature when reconstructing past environments is authentic illumination citeDC01). Today the interior of our buildings are lit by bright and steady light, but past societies relied on daylight and flame for

illumination. Our perception of an environment is affected by the amount and nature of light reaching the eye. A key component in creating the authentic and engaging virtual environments will be the accurate modeling of the visual appearance of the past environment illuminated by flame.

9 Authentic Illumination for Viewing Cultural Heritage Sites

For the dark interior of ancient environments, some form of firelight was necessary. The fuel used for this fire directly affects the visual appearance of these interiors and any objects within them citeDC01. Furthermore, flames are not static and their flicker may create patterns and moving shadows that further affect how objects lit by these flames might look. Thus any realistic flame-lit environment should not only incorporate the accurate spectral profile of the fuel being burnt, but also the manner in which the flame may move over time [Rac96, RC03, BLRR06]. The acquisition of valid experimental data is thus of vital importance as the material used may have had a significant influence on the perception of the ancient environment [DCB02]. Experimental archeology techniques should be used to build a variety of reconstruction candles and oils for lamps using authentic materials. Figure 39 shows some reconstructed ancient light sources, including, processed beeswax candles, a tallow candle (of vegetable origin), an unrefined beeswax candle, s selection of reeds coated in vegetable tallow, rendered animal fat lamp, and an olive oil lamp.

Figure 39: Experimental archeology: reconstructing ancient light sources

Detailed spectral data of each flame type can then be gathered using, for example a spectoradiometer. Such a device is able to measure the emission spectrum of a light source from 380nm to 760nm, in 5nm wavelength increments. This gives an accurate breakdown of the emission lines generated by the combustion of a particular fuel [RC03]. This wavelength information can subsequently be incorporated into a physically based renderer.

10 Recreating the Prehistoric Site of Cap Blanc

The Upper Palaeolithic cave art of Franco-Cantabria is a remarkable record of the earliest representational artistic expression, consisting of images of the ice age European fauna, signs and handprints. The rock shelter site of Cap Blanc, overlooking the Beaune valley in the Dordogne, contains perhaps the most dramatic and impressive example of Upper Palaeolithic haut-relief carving. A frieze of horses, bison and deer Ů some overlaid on other images Ů was carved some 15,000 years ago into the limestone as deeply as 45cms, covering 13m of the wall of the shelter. Since its discovery in 1909 by Raymond Peyrille several descriptions, sketches, and surveys of the frieze have been published, but they appear to be variable in their detail and accuracy. Figure 40 shows a photograph of part of the horse frieze.

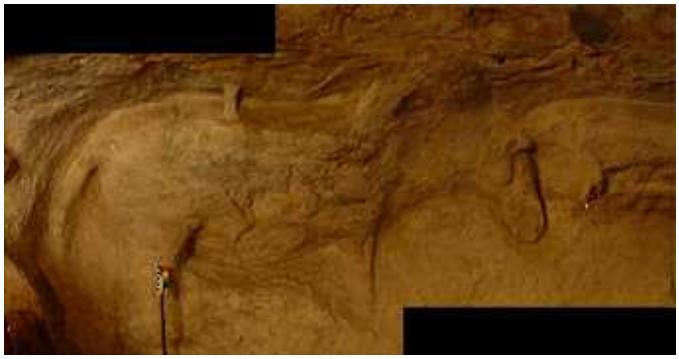


Figure 40: Photograph of part of the horse frieze from Cap Blanc

Developing an understanding of the human perception of these caves should make a significant contribution to our broader understanding of the significance of parietal art during this period. A key question to consider is whether the dynamic nature of flame, coupled with the careful use of three-dimensional structure of the carvings, may have been used by our prehistoric ancestors to create animations, rather than only static images, in the cave art sites of France, some 15,000 years ago. A virtual environment enables the chamber context and illumination to be considered in a safe and controlled manner. The fidelity of this virtual reconstruction is paramount if we are to avoid misleading representations of carvings under flame light.

Figure 41 shows the horse illuminated by a simulated 55W incandescent bulb (as in a low-power floodlight), which is how visitors may view the actual site today. In Figure 42 the horse is now illuminated by a simulated animal fat tallow candle as it may have been viewed 15,000 years ago. As can be seen the difference between the two images is significant with the candle illumination giving a "warmer glow" to the scene as well as more shadows. This shows that it is important for archaeologists to view such art work under (simulated) original conditions rather than under modern lighting. (It is of course impossible to investigate these sensitive sites with real flame sources).

Figure 41: Virtual reconstruction lit by 55W incandescent bulb

Figure 42: Virtual reconstruction lit by animal fat candle

Such a high-fidelity environment does open up some tantalizing possibilities. We will of course never know for certain whether the artists of the Upper Palaeolithic were in fact creating animations 15,000 years ago, however the virtual environment does show that it certainly is possible. There is other intriguing evidence to support this hypothesis. As can be seen in the figures, the legs of the horse are not present in any detail. This has long been believed by archaeologists to be due to erosion, but if this is the case, why is the rest of the horse not equally eroded? Could it be that the legs were deliberately not carved in any detail to accentuate any motion? Furthermore traces of red ochre have been found on the carvings. It is interesting to speculate again whether the application of red ochre at key points on the horse's anatomy may also have been used to enhance any motion effects. It is possible to investigate all this in a high-fidelity virtual environment.

Part V

Speeding up Global Illumination

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 synthesized 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 modeled 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.

11 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 localization, 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, CCM03]. 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.

11.1 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, labeled 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, WFM01].

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].

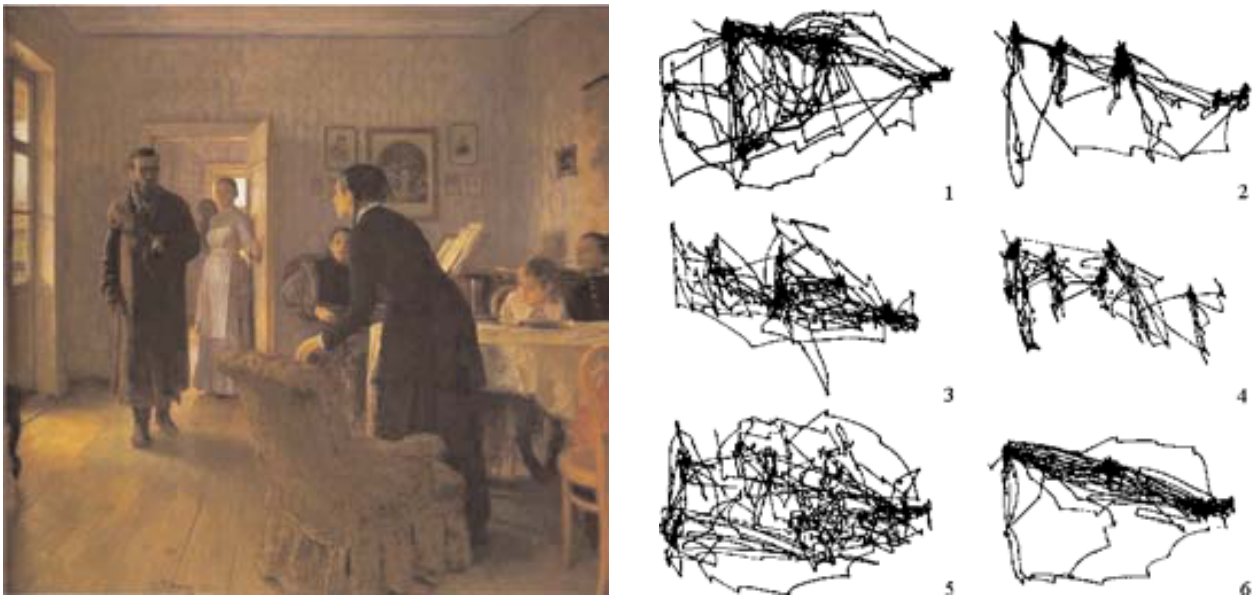
11.2 Peripheral Vision

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. [WFM97] 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.

11.3 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 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 43.



(a) Repin's picture

(b) Eye movements

Figure 43: 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]

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].

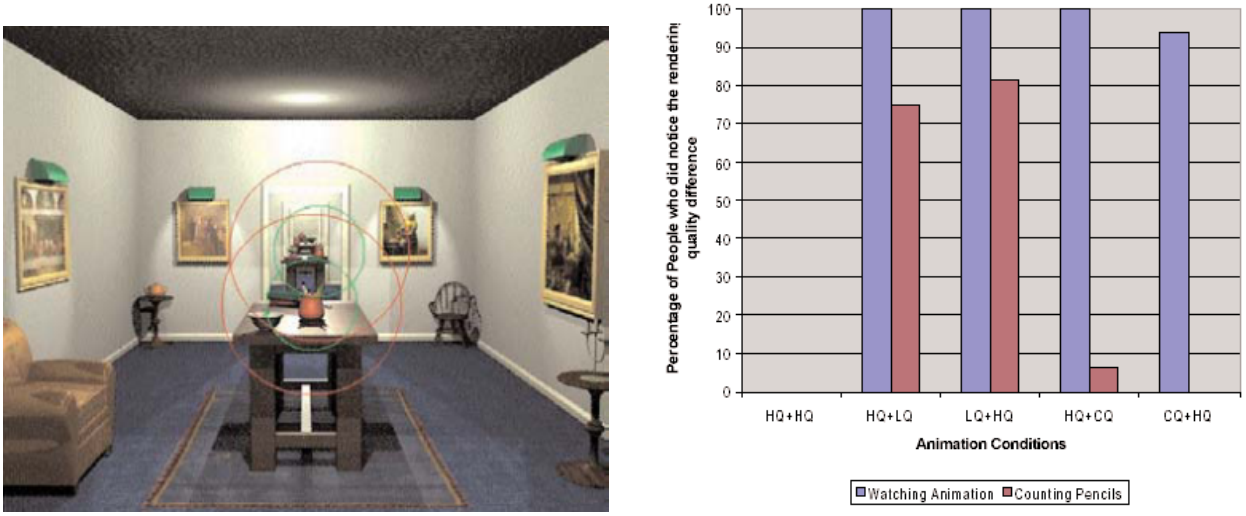
Previously, Cater et al. [CCL02] showed that the conspicuous objects in a scene that would normally attract the viewer's attention can be deliberately ignored if they are irrelevant to the task at hand. The effectiveness of inattentional blindness in reducing overall computational complexity was investigated 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) centered around the pencils, shown by the inner green circle in Figure 44(a). The high quality is blended to the low quality at 4.1 degrees visual angle (the outer red circle in figure 44(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 the animations. To minimize experimental bias the choice of condition to be run was randomized 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 color, 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.



(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 44: 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

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 color of the mug (90%) which was in the foveal angle and even less the correct color 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 color 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.

11.4 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 color, 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, color, 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.

11.5 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:

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.)

- 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.

11.6 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.

11.7 Map Generation

The process begins with a rapid image estimate (in the order of ms) of the scene using a quick rasterization 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 45 shows the various maps for the reference image 47(a). Sub-figures 45(a) and 45(b), in Figure 45 show the task map with and without a foveal-angle gradient added. Sub-figure 45(c) demonstrates the saliency map, while 45(d) shows the combination of the task map and the saliency map with equal weighting.

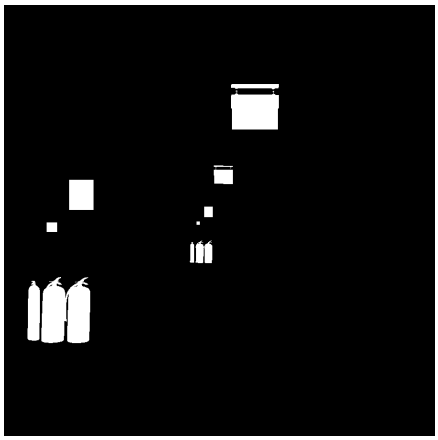
11.8 Results

To test their importance maps, Sundstedt et al. rendered ten animations of a corridor scene [SDL*05] shown in Figure 47. 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 46 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 frame to the same detail as the task objects in Sub-figure 45(a) takes 5.2 times longer than the optimized 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].

12 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



(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 45: Importance map examples from the corridor scene

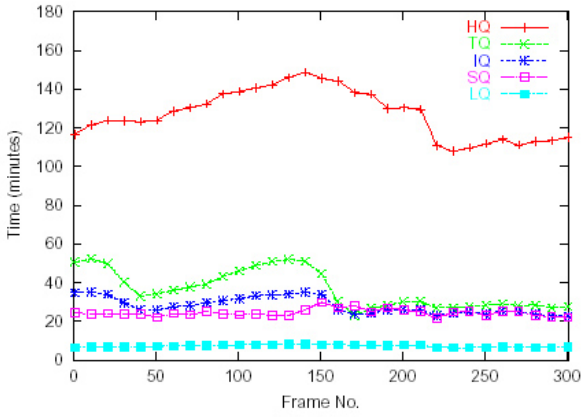


Figure 46: Timing comparison for corridor A between the reference high quality solution (HQ) and the different conditions generated using the *importance map*

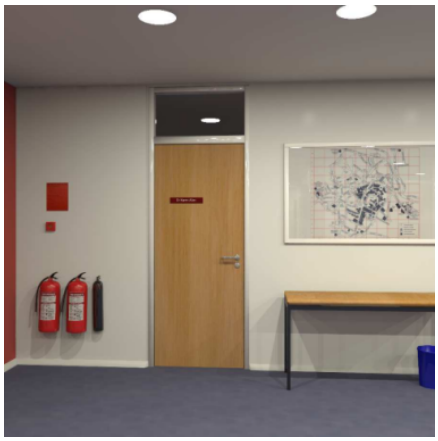
(a) Corridor A: The first frame

(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 47: Corridors A and B from a 312 frame animation

prioritize 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.

Acknowledgments

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.

Part VI

Rendering Nature

Simulating natural phenomena has always been one of the most challenging goals of computer graphics. In particular, the effect of the atmosphere on sunlight, under certain circumstances, causes a number of really spectacular phenomena; some are well-known, like the mirages, whereas some of them remain mostly unknown, such as the Fata Morgana or the Novaya-Zemlya. But these effects go beyond mere beautiful images; some of them offer the viewer cues regarding important aspects such as scale (aerial perspective) or temperature: the sensation of heat in a road will be unconsciously magnified by adding a mirage on the road, or by letting the horizon flicker even subtly. Therefore, modeling optical effects in a virtual environment not only enriches its depiction, but they offer the opportunity for greater fidelity as well, beyond the visuals.

Unfortunately, simulating correctly these effects is a costly process, and real-time is currently out of the question. We can cheat the physics of the system, applying ad-hoc methods and cutting corners here and there, but our system will then lose predictability. In order to come up with a high-fidelity virtual scenario we need to base it on the laws of physics! What we can do to reduce rendering times is to use perceptual issues, based on our knowledge of how the human visual system works. The previous section of this course §11 discussed all the aspects regarding perceptual issues; we will now discuss several atmospheric effects, what causes them and how they can be simulated.

13 Light in the Atmosphere

Most of the global illumination algorithms consider that light is propagated following straight lines. This is so because these algorithms presuppose either that there are no media through which the light travels, or that these media are homogeneous, that is, they have the same properties in all their points. But the atmosphere is in fact a non-homogeneous medium, with its properties varying continuously from point to point, and this affects the path that the light follows through it. As we will see, the most important property in terms of defining the light paths (which will yield the effects shown in this section) is the index of refraction, which in turn depends strongly on temperature, density and wavelength (and more subtly on other parameters such as humidity). The continuous variation of the light paths induced by the non-homogeneous index of refraction can be described by a differential equation, as will be shown later.

13.1 Previous Work

Berger and Trout [BTL90] are the first to observe that the ray tracing method only changes the trajectory of the ray when it intersects an object. Therefore, the slight changes in direction that rays experiment when going through the atmosphere were not considered. Those changes are, precisely, the origin of the mirages. According to Fermatt's principle, light crossing a medium gets curved towards the areas with a greater index of refraction. The index of refraction of the air depends on both humidity and density, but the effect of the former on light is very small and can be ignored. On the contrary, density, which is a function of pressure and temperature, can significantly change the light rays trajectories. In the case of the mirage effects, these occur in a very small altitude range, where pressure is practically constant; therefore, the change in the index of refraction can be supposed to be owed only to temperature variations. In their work, the authors only deal with situations in which the index of refraction depends solely on height, and solve the problem dividing the medium into various homogeneous layers. The proposed solution is to subdivide the atmosphere in multiple horizontal layers with a different index of refraction for each one. When using ray tracing, the ray launched from the observer is refracted every time it crosses a boundary between layers. This method is then only valid for situations where the index of refraction is only a function of the height; on the other hand, it is impossible to obtain the real curved path followed by the ray, coming up instead with an approximation made of straight segments.

Musgrave [Mus90] points out a few considerations regarding the interpretation of the model in [BTL90] and its implemented version. He indicates that the primary bending agent in the formation of mirages is total reflection, instead of refraction as [BTL90] propose. As a consequence, a purely reflective model might suffice, without the need for refraction. Musgrave reinforces this idea by stating that, if refraction were the primary engine in mirage formation, one would then expect to see the effects of dispersion in a mirage, smearing the image into its component colors. This is in fact a key factor in the green flash effect during sunsets [Min93]. On the contrary, there is little to no dispersion in mirages observed in nature.

Stam and Languenou [SL96] present their work on ray tracing in non-constant media. They do not use Snellt's law to calculate the trajectory of the light ray, unlike the previous works described. On the contrary, they obtain the differential equation that describes the trajectory of the ray from the equation of the light wave. The authors also observe that the index of refraction of the atmosphere is inversely proportional to the temperature of the air, with an equation that relates both magnitudes, closely matching experimental data. For a given hot horizontal surface at a temperature T_s, the temperature above it has an exponential falloff characterized by the attenuation distance d_0. If the attenuation distance is big enough, the rays follow approximately a parabolic trajectory, and the method described in [BTL90] yields good results. However, if the attenuation distance is small, the trajectories become hyperbolic, and the approximation in [Mus90] works better by using total reflection. To solve their equation [SL96] propose a perturbation method [Kra90], and applies it to two particular cases. The first one consists of a superposition of N hot nuclei, whereas in the second one the distributions of the index of refraction are defined by a stochastic model, that is, a Fourier transform of a random index of refraction.

On the other hand, Gröller [Grö95] considers different situations that cause non-linear paths. These can be the result of underlying dynamic systems that act upon a particle, such as gravity centers, gravity lines or dynamic chaotic systems, or can be defined arbitrarily in an explicit

manner through equations. In the first case, the curved rays are the solutions of first-order ordinary differential equations, although usually the exact solution cannot be calculated analytically. It is then necessary to use numerical techniques, and he chooses Euler's method to calculate successive points of the curve path knowing the position and the tangent vector at the previous point. Another problem, inherent to curved ray tracing and light, is obtaining the trajectory that goes from the intersection point to a light source. Here, the author proposes two possible solutions. The first one is supposing that the light rays are only curved from the intersection point to the eye, but travel in a straight line from the light source to the intersection point. The second one is assigning color to the intersection point regardless of the light sources, for instance by using textures with the illumination baked-in. A third solution involves voxelizing the space and precomputing incoming curved light trajectories, to be used when shooting a shadow ray.

13.2 Curved Paths

When light passes from one homogeneous medium to another with a different index of refraction, its trajectory is modified according to Snell's law, which gives the relationship between angles of incidence and refraction, with the ray being continuous across the boundary:

$$n_1 \sin\theta_i = n_2 \sin\theta_t \tag{5}$$

where n_1 and n_2 are the indexes of refraction of the first and second medium respectively, θ_i is the incidence angle with respect to the normal to the boundary surface and θ_t is the angle of the refracted ray with respect to that normal. The index of refraction is defined as the quotient between the speed of light in that medium and the speed of light in a vacuum:

$$n_i = \frac{v_i}{c} \tag{6}$$

This phenomenon is known as refraction, and it is traditionally accounted for in any ray tracer. It shows how light gets bent towards the medium with the greater index of refraction.

Snell's law is formulated considering two homogeneous media, through which light travels in a straight path. But all media are inhomogeneous at some degree, with their different properties varying from point to point, and the atmosphere of course is no exception. When the medium is inhomogeneous and the index of refraction changes continuously from point to point, the trajectory of light is affected at each differential step, not just at the boundaries between two media. The final result of this differential change is a curved path that distorts the normal view of a real scene, but that path cannot be calculated by traditional ray tracing. We apply Fermat's principle [Gla95b] to obtain the curved trajectory of the light rays. The original statement of Fermat's principle is "the actual path between two points taken by a beam of light is the one which is traversed in the least time." A reformulation of the principle can be expressed as "light, in going between two points, traverses the route l having the smallest optical path length Λ". This optical path length Λ is defined as the distance traveled in an optical system by light. For a series of continuous layers with index of refraction n as a function of distance l, it is given by

$$\Lambda = \int_A^B n_\lambda \, dl \tag{7}$$

where n_λ is the index of refraction as a function of l and l is the distance. As a matter of fact, Snell's law is just a particular case of this principle [BW02]. In its differential form, it can be written as:

$$d\Lambda = n_\lambda \, dl \tag{8}$$

Under standard circumstances, the effects are negligible. It is under non-standard circumstances when we really see some amazing atmospheric phenomena. Probably the best known is the inferior mirage, which occurs when lower regions of the atmosphere are much hotter than higher regions, as in deserts. Since the index of refraction is an inverse function of temperature, light beams get bent upwards, and what we see on the ground is actually the image of objects above it. The effect is also known as the road mirage, since it is very usually seen in roads when the asphalt is much hotter than the surrounding air. Figure 48 graphically explains this effect occurs, along with a real picture of the phenomenon.

Fermat's principle has been recently reformulated as "a light ray, in going between two points, must traverse an optical path length which is stationary with respect to variations of the path", as described by Glassner in [Gla95b]. The optical path length presents a stationary value whenever $\delta\Lambda = 0$, therefore corresponding to a maximum, minimum or saddle point. Taking equation 7, $\delta\Lambda$ is given by:

$$\delta\Lambda = \delta \int_A^B n_\lambda \, dl = \int_A^B \delta n_\lambda \, dl + \int_A^B n_\lambda \, \delta(dl) = \int_A^B \frac{\partial n_\lambda}{\partial x_i} \, \delta x_i \, dl + \int_A^B n_\lambda \, \delta(dl) \tag{9}$$

where x_i are the components of l. Given that $dl^2 = dx^2 + dy^2 + dz^2$, considering dx_i as variables and taking increments we get:

$$\delta(dl) = \frac{dx_i}{dl} \, \delta(dx_i) \tag{10}$$

Figure 48: Left: explanation of the origin of an inferior mirage. Right: picture of a real inferior mirage, also known as road mirage.

so the second integral in equation 9 becomes:

$$\int_A^B n_\lambda \, \delta(dl) = n_\lambda \frac{dx_i}{dl} \delta x_i \Big|_A^B - \int_A^B \delta x_i \frac{d}{dl}\left(n_\lambda \frac{dx_i}{dl} \right) dl \tag{11}$$

Since the different considered trajectories start in the fixed points A and B, $\delta x_i(A) = 0$ and $\delta x_i(B) = 0$, so equation 11 results as follows:

$$\delta\Lambda = \int_A^B \left[\frac{\partial n_\lambda}{\partial x_i} - \frac{d}{dl}\left(n_\lambda \frac{dx_i}{dl} \right) \right] \delta x_i \, dl = 0 \tag{12}$$

This equation must be true for any value of δx_i, which lets us come up with the equation to obtain the trajectory of a light ray in an inhomogeneous medium with a known index of refraction, which is:

$$\frac{d}{dl}\left(n_\lambda \frac{d\vec{r}}{dl} \right) - \nabla n_\lambda = 0 \leftrightarrow \frac{d}{dl}\left(n_\lambda \frac{dx_j}{dl} \right) - \frac{\partial n_\lambda}{\partial x_j} = 0 \tag{13}$$

where l is the length of the arc, n is the index of refraction of the medium and with $(j = 1,2,3)$ are the coordinates of the point. Since the index of refraction n can be obtained for each point in space (see section §13.3), the equation can be solved numerically. Naive resolution methods will yield huge computation times, so we need a robust, efficient numerical method. In these notes we apply the RK5(4)7M embedded Runge-Kutta method, from the Dormand-Prince family (see [BF88] and [DP80]). A detailed description of the implementation is provided in Appendix A of this chapter, along with an analysis of the error and computation times.

13.3 Atmosphere Profiles

To model the atmosphere we start with the 1976 US Standard Atmosphere [USG76]. It is an average of pressure and temperature for various altitudes, which make up the ideal, steady state profile. The values selected in 1976 come from slight modifications of the values adopted in 1962.

Figure 49 shows the temperature and pressure values for the first $34Km.$ of standard atmosphere from the values of the Standard Atmosphere (intermediate values are obtained by interpolation). As it can be seen, temperature decreases with height at an approximate rate of $0.6^{O}C$ per $100\,m.$ Under these conditions, air is perfectly stable and no unusual effect takes place[1].

Starting with the standard values, we seek to obtain a profile of the index of refraction for any arbitrary atmosphere and as a function of wavelength, in order to render the desired effects. We first need to obtain the density profile, for which we take the temperature and pressure data and apply the Perfect Gas Law:

$$\rho(h) = \frac{P(h)\,M}{RT(h)} \tag{14}$$

where ρ is the density we want to obtain, T is the temperature, P is the pressure, M is the mean mass of the molecules of a mixed atmosphere ($28,96 \cdot 10^{-3}\,kg/mol$ typically) and R is the gas constant, $8,314510\,J/mol \cdot K$.

[1] Actually, the sole presence of the atmosphere itself causes effects such as the astronomic refraction, which needs to be corrected for to accurately record the position of stars [You00]; additionally, an ideal stationary state is never achieved, which reflects in the twinkling of distant lights at night.

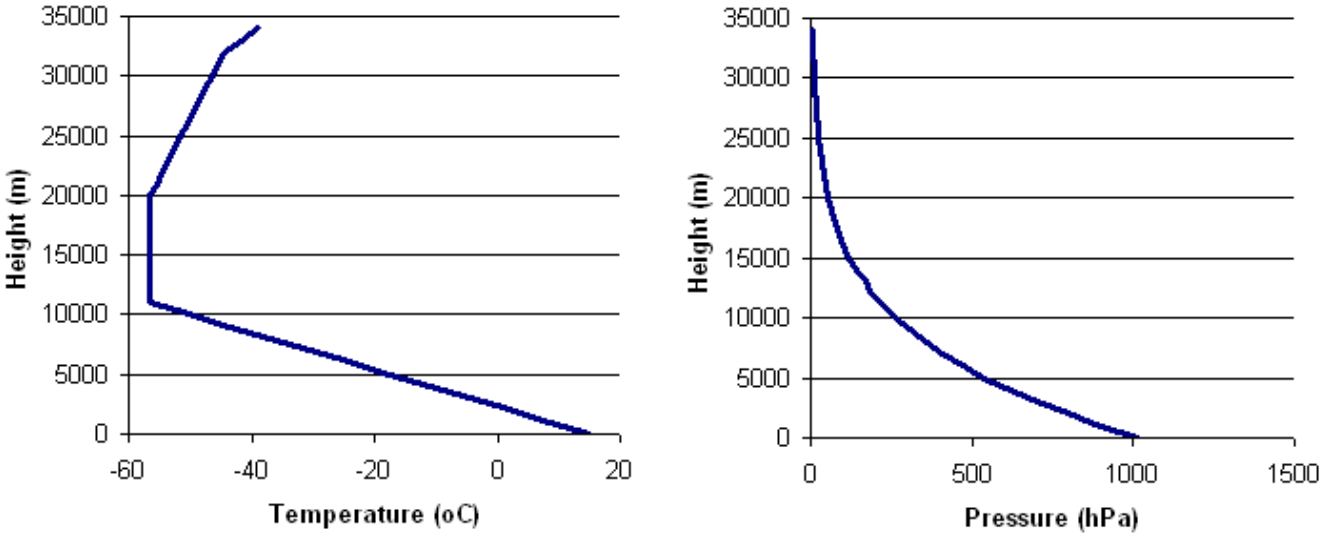


Figure 49: Left: temperature profile of the USA 1976 Standard Atmosphere. Right: pressure profile of the USA 1976 Standard Atmosphere.

We now need an expression for the index of refraction in a medium as a function of wavelength. For low-density media (such as air), we can take Cauchy's dispersion formula [BW02]:

$$n(\lambda) = a \cdot \left(1 + \frac{b}{\lambda^2}\right) + 1 \tag{15}$$

where a and b are constants which depend on the medium. In case of air $a = 28,79 \cdot 10^{-5}$ and $b = 5,67 \cdot 10^{-5}$. This formula yields good approximations between the observed and the calculated $n(\lambda)$ for air, with errors of the order of 10^{-3}. Figure 50 shows a comparison between actual measured data and the values predicted by the formula.

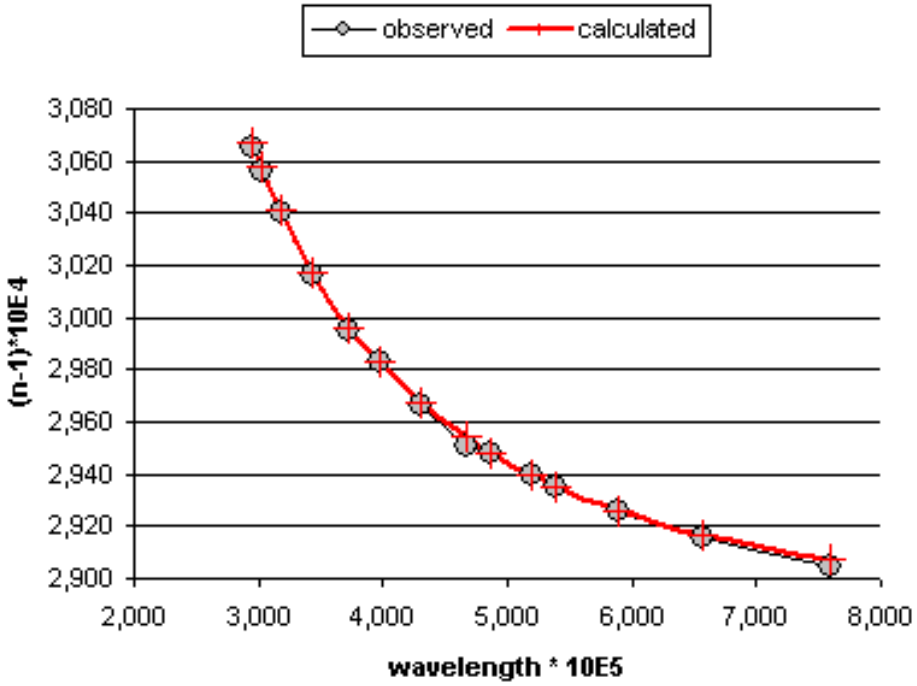


Figure 50: Observed values of $n(\lambda)$ for air, and the values predicted by equation 15 (Observed data after [BW02])

At this point we have on the one hand density as a function of height (equation 14) and on the other hand the index of refraction n as a function of wavelength (equation 15). To obtain n for any height and any wavelength we should combine both equations, which can be done by applying the Gladstone-Dale formula [GD58]. The formula describes how refractivity (refraction index minus 1) is proportional to the density according to the following equation:

$$n(h,\lambda) = \rho(h) \cdot (n(\lambda) - 1) + 1 \tag{16}$$

Equation 16 is the final expression which allows us to describe an exact profile for the index of refraction in any atmosphere, as a function of wavelength and starting with pressure and temperature data. The effect of humidity can easily be incorporated to our model, in which case the index of refraction can be obtained by using the formula proposed by Ciddor in [Cid96], and using the Modified US 1976 Standard Atmosphere, as proposed by Van der Werf [vdW03]. A full schema of the derivation of the atmosphere model can be seen in Figure 51. It includes the Atmospheric Profile Manager, described in the following subsection, which will be used to de-standardize the atmosphere in order to recreate the desired phenomena.

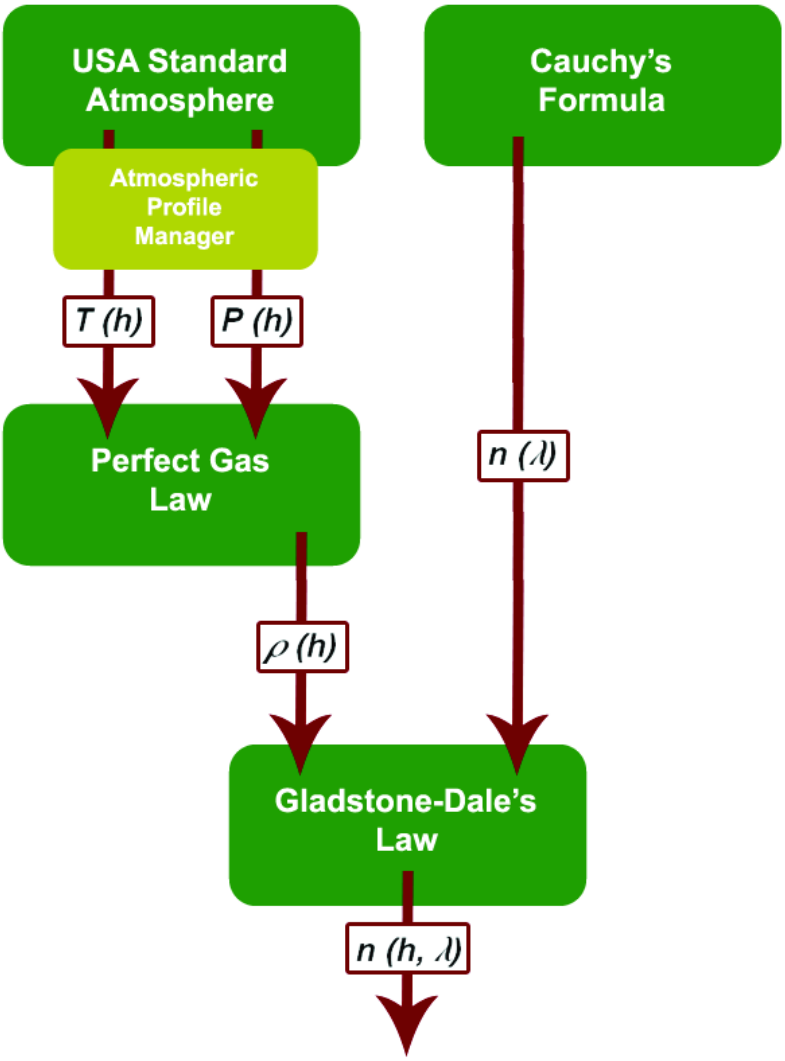


Figure 51: Diagram of the process to obtain the index of refraction of the atmosphere.

13.4 De-Standardizing the Atmosphere

A description of a standard atmosphere is a necessary starting point in that it ensures that the simulation is based on actual data. The next step in the simulation process is to de-standardize that atmosphere, recreating the conditions necessary for the effects to occur. To do that we have developed the Atmospheric Profile Manager (APM), actually just a catchy name for a little piece of code which enables to set different temperature and pressure profiles at different heights. From there, a new profile for the index of refraction is obtained as explained in the

previous subsection (see again Figure 51), and the resulting curved light paths will yield the desired images. Since the non-standard profiles for atmospheric phenomena generation are always close to ground level (with a maximum altitude of $500\,m$), we consider the rest of the atmosphere to remain standard, thus speeding up calculations.

13.4.1 Inversion Layers

Most mirages and similar atmospheric phenomena are due to inversion layers. In the normal situation, the temperature decreases with height during the first thousand meters but under the right conditions inverted temperature gradients are formed. These inversion layers are characterized by the height at which they occur and their inverse temperature gradient. To insert them into the standard profile, thus altering it, we need a method suitable for ray tracing which also offers enough flexibility to tweak the parameters describing the layers. We use an analytical function known as the Fermi distribution, as proposed by Van der Werf et al. in [vdWKL03] (see equation 17):

$$T(h,x) = T_{ATM}(h) - \Delta T(x) + \frac{\Delta T(x)}{1 + e^{\frac{h - h_{ciso}(x)}{a(x)}}} \tag{17}$$

where x is the distance in the direction parallel to the Earth's surface, $h_{ciso}(x)$ is the height of the inversion layer about which the added temperature profile is centered, $\Delta T(x)$ is the temperature jump across the inversion and the diffuseness parameter $a(x)$ determines the width of the jump (see Figure 52). This distribution yields very accurate results, as can be seen in another work by Van der Werf in which the author uses the same techniques [vdW03] and compares the predicted atmospheric profiles with well-established standards such as the Star Almanac Atmosphere [SAA51], the the Nautical Almanac Atmosphere [NAA58] and the Pulkovo tables [V.K85]. Figure 53, top right, shows a proof-of-concept rendering of the distortion caused by an inversion layer.

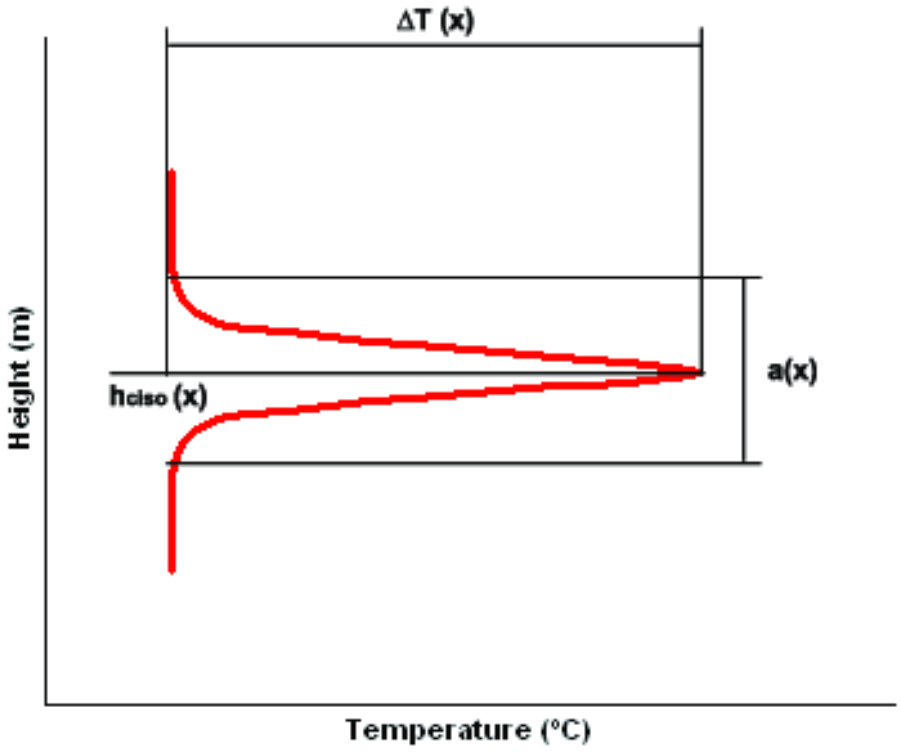


Figure 52: Graph of the application of an inversion layer.

The Fermi distribution is also suitable for animation, by allowing several parameters to be key-framed:

- $h_{ciso}(x)$ can be modified by a certain time-based function $f(t)$ by the additive expression $h_{ciso}(x,t) = h_{ciso}(x) + f(t)$. Animating this parameter shifts the mirage vertically.
- $a(x)$ can be scaled by another time-based function $f'(t)$ by using the expression $a(x,t) = f'(t)a(x)$. This parameter can soften or sharpen the mirage.

- $\Delta T(x)$ can be scaled also by a factor $f(t)$ by the expression $\Delta T(x,t) = f(t)\Delta T(x)$. This is maybe the most interesting parameter to animate, as it can show how the simulated effects form over time.

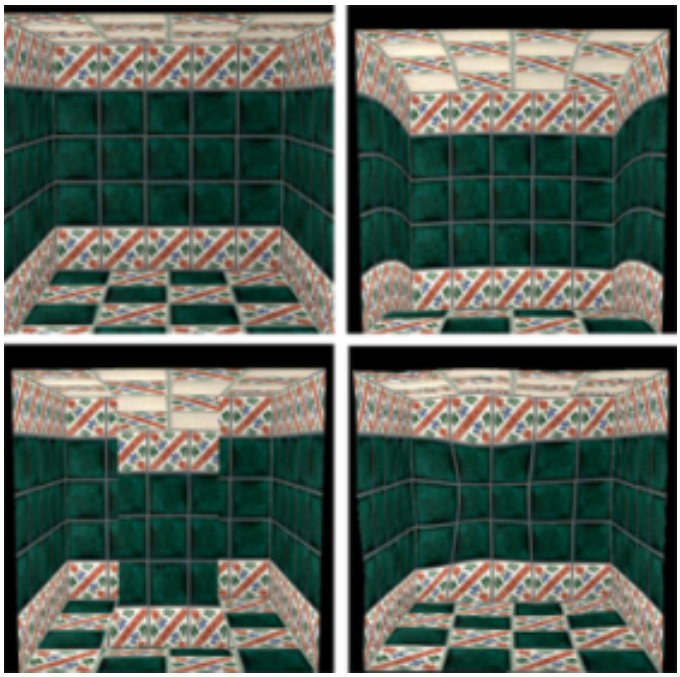


Figure 53: Concept pictures of the phenomena being simulated by de-standardization methods. Left: inversion layers. Middle: hot spots. Right: noise grids

13.4.2 Hot Spots

Some effects can be bound to some local region of space. In the case of a road mirage this region would be the asphalt of the road itself (much hotter than its surroundings). To allow for that type of situations, we define a *hot spot* as an increase in temperature at a certain position in space, which affects its nearby region. To obtain the new atmospheric profile as influenced by the hot spot we first need to obtain the new temperature distribution created by it; we use the formula proposed by Stam [SL96] (equation 18), which is based on experimental data found in the optical sciences literature [Owe67]:

$$T = T_0 + (T_s - T_0)\exp(-d/d_0) \tag{18}$$

where T_s is the known temperature of the hot spot, d is the distance to the hot spot, $T_0 = 273K$ and d_0 is the drop-off length (for the example mentioned above, the drop-off length for the heat field generated by a hot asphalt road is approximately one centimeter, as stated in the book by Minnaert [Min54]).

The concept of *hot spot* can be extended to other geometric elements, like lines, planes and even curves, surfaces or volumes. The only variation on equation 18 would be the substitution of d by a more complex expression of the distance to the chosen element; however, experience says that hot spots are usually enough for modeling all localized effects. Therefore, our current version of the APM only implements hot spots, although it can be easily extended in the future to other geometric elements. The application of the localized effect can seen in Figure 53, bottom left. This definition of a hot spot is also suitable for animations.

13.4.3 Noise Grids

Noise grids have also been added to the APM. They enable the simulation of effects caused by unstable heat waves, coming for instance from a hot car engine or an activated jet engine (see Figure 53, bottom right). Noise grids are 3D regions of space where temperature varies randomly between a defined maximum and minimum. A random ΔT value is set at each vertex of the grid, and the values inside the grid are interpolated by Hermite interpolation. The effect can be animated by adding time as a fourth dimension to the grid. The result is similar to the rippling effect caused by extremely hot surfaces.

Effects generated by noise grids could also be generated by a set of hot spots, but they are a very useful tool in the APM for fast modeling of atmospheric profiles for certain effects. A combination of both is also possible: for instance, to simulate distortions caused by fire, a noise grid can provide the general turbulence whereas hot spots can keep the effect localized to those parts of the scene close to the fire. As stated

at the beginning of the chapter, these type of distortions not only add to the realism of the scene, but they provide important cues regarding temperature.

14 Ghost Ships and Fairy Tales: Mirages and Other Optical Effects

We are now going to bend light to obtain rendered images of some of the spectacular effects that can be seen in nature. For each one, we will first introduce a bit of the history and legend behind it, then offer the scientific explanation and render the corresponding image.

We have designed scenes with very simple geometry, but with very accurate atmospheric profiles and using real-world distances and dimensions. The Earth is always modeled as a sphere of 6375 units of radius (obviously one unit in the 3D scene equals one kilometer in the real world). A slightly bigger, concentric sphere forms a 40-kilometer atmosphere layer. The rest of the setup of the scenes is characteristic of each different simulated effect. The simplicity of the whole setup is irrelevant, though, since it is the curvature of the ray traversing the medium the only factor that matters. This correct curvature is guaranteed by a) a precise atmosphere profile, b) using real geometric dimensions and c) accurately solving the physically-based equation that governs the paths of the light rays.

14.1 Inferior Mirages

The inferior mirage, also known as the "water on the road" effect, is probably the best known atmospheric effect owed to curved light paths[2]. This effect occurs when there is a strong decrease of temperature with increasing height. As a consequence, the light rays approaching the ground are curved upwards, generating an inverted image of the object in the ground. The inferior mirages tend to shake and change rapidly since hot air is less dense than cold air, and it is therefore too unstable to remain static. Contrasting with this, superior mirages (see section §14.2) are very stable since a denser cold mass at the bottom will remain mostly static.

Figure 54 left shows a picture of a real inferior mirage. Figure 54 right shows a simulation of the same phenomenon (differences between the two images are owed to the lack of accurate data when the picture was taken). The shuttle seems to be reflected as if the ground level was a mirror. The temperature profile used for the rendered image can be seen in Figure 55, while Figure 56 shows an animation of the inferior mirage as it is formed. Randomly jittering the parameters in the APM will give the animation a very realistic hot-ground effect. However, for the purpose of virtual environments where rendering times must be kept within interactive frame rates, the results of the first simulation could be precomputed and cached, jittering them randomly in subsequent frames for a very realistic effect.

Figure 54: Left: Real picture of an inferior mirage. Right: Rendered inferior mirage.

14.2 Superior Mirages: The Ghost Ship

In 874 the Norse Vikings settled in Iceland. A century later, a former chief named Erik the Red was banished from Iceland, and he and his crew set sail around 982 to find another place to live. They ended up in Greenland, which surprisingly is not visible from Iceland owed to the Earth's curvature... What made him head in that direction[3]?

Given the right conditions, it is possible for objects lying beyond the horizon to actually be seen [SL76], in a manifestation of the superior mirage. The object itself will remain hidden by the Earth's curvature, but its mirage will be seen above the horizon. Also known by its Icelandic word as the hillingar effect, the superior mirage occurs when there is an increase of temperature with height near the ground, meaning there is an inversion layer. As a consequence, the light rays approaching the ground are curved downward. They are also known as arctic mirages, since the images are widely seen in arctic latitudes.

Depending on the gradient of the inversion layers, the arctic mirage can offer an inverted or an upright view of the real object. Figure 57, left, shows a real picture of a ship, with the simulated image on the right[4]. Again, some discrepancies are to be expected owed to lack of reliable data. The simulation was done with the ship located at 2.6 km from the viewer, with the camera at 10 meters above ground level. The temperature profile can be seen in Figure 58. Figure 59 shows the superior mirage as it forms over time.

[2] The word *mirage* comes from the Latin *mirari*, 'to be astonished'

[3] Other historical references regarding superior mirages can be found in [Hei]

[4] It is interesting to note how this could explain the countless ghost ship stories told centuries ago by highly superstitious mariners.

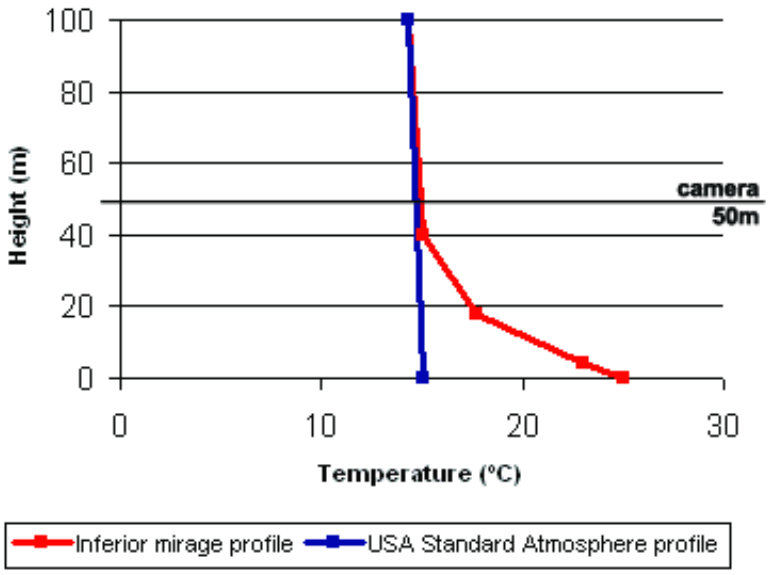


Figure 55: Temperature profile of the inferior mirage rendered in figure 54 compared to the USA Standard Atmosphere's temperature profile.

Figure 56: Frames of an animation of the inferior mirage.

14.3 The End of the World?

We continue our exploration with the ancient arctic legend of the Viking's end of the world. They believed that there lay a land at the end of the world, a land they called Ultima Thule, where all ocean waters flowed back to the known lands. According to several sources, the word Thule comes from the Celts, and means "to raise oneself". Vikings also believed in massive sea walls that would raise over the horizon as one approached Ultima Thule [Mal03].

Maybe the legend was born from a very scientific fact: the distortion of the horizon which occurs under very common polar atmospheric conditions. When the temperature of the lower atmosphere rises at a rate of 11.2C per 100 meters, the light rays bend in an arc exactly equal to the curvature of the Earth, and the horizon will appear flat, like an infinite plane. If the inversion gradient becomes even stronger, the light rays exceed the curvature of the Earth, and the horizon will appear to rise vertically from the flat position.

To recreate this, we have covered the Earth sphere with a tiled texture of the Northern Bering Sea, and given the atmosphere sphere a grayish

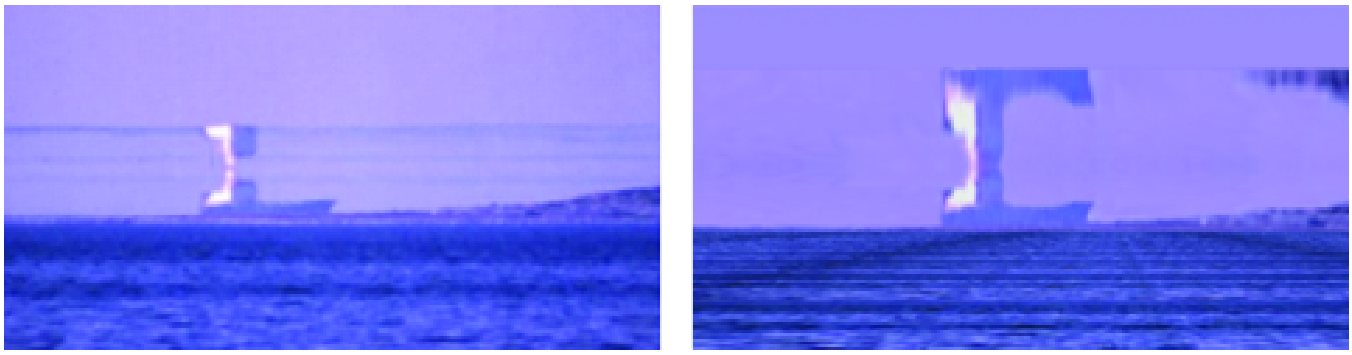


Figure 57: Left: Real picture of a superior mirage. Right: Rendered superior mirage.

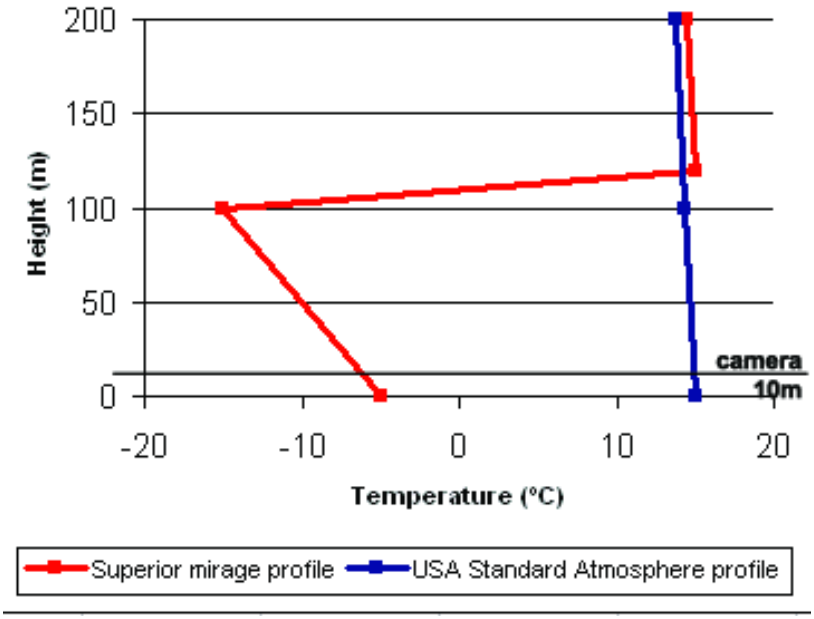


Figure 58: Temperature profile of the superior mirage rendered in figure 57 compared to the USA Standard Atmosphere's temperature profile.

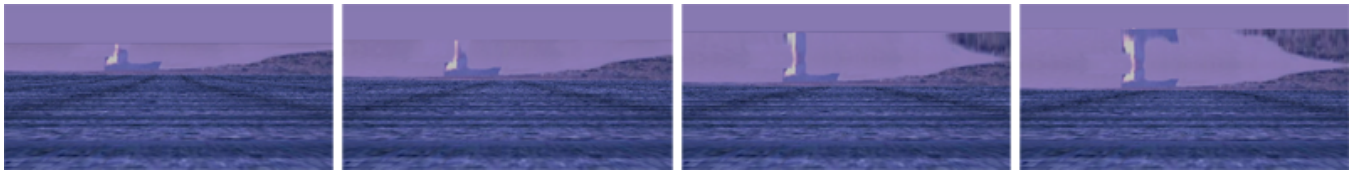


Figure 59: Frames of an animation of the superior mirage.

sky color. The index of refraction varies continuously with height, with a temperature profile as seen in 61. Figure 60 (left) shows the scene rendered with a standard ray tracer: no curvature at all has been applied to the rays, and therefore the horizon appears undistorted. Figure 60 (right), on the other hand, shows how the horizon gets in fact distorted when taken into account the real paths that light travels in a polar atmosphere. The horizon line clearly rises above its real location, while getting slightly curved upward.

Figure 60: Left: Northern Bering Sea, rendered without ray curvature. Right: Rendered image of the distortion in the horizon caused by a polar atmosphere.

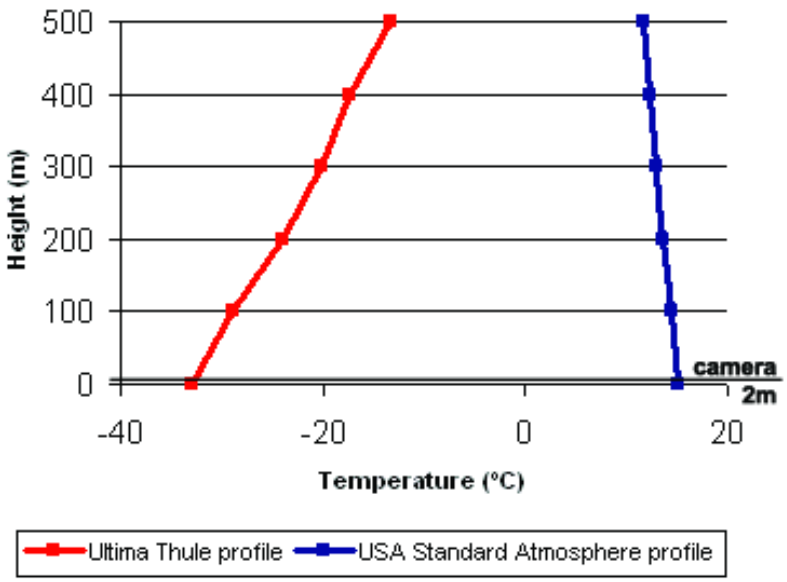


Figure 61: Temperature profile of the Viking's end of the world effect compared to the USA Standard Atmosphere's temperature profile.

14.4 Fata Morgana

Fata Morgana is the name of a fairy enchantress skilled in the art of changing shape, and literally means "the fairy Morgan" in Italian. One of the versions of the legend makes her King Arthur's sister and explains how she learned many of her skills from Merlin the Magician. She lived in a castle under the sea, and sometimes she would make her castle appear floating in the sky [TGD68].

The mirage known as Fata Morgana occurs when there are several alternating cold and warm layers of air near the ground or the sea. These alternating layers cause a multiple concatenation of inferior and superior mirages, resulting in a complicated superposition of images of the same object (Figure 62 left). The number of inversion layers usually varies between two and five, with the images alternating between upright and inverted. In addition to this, slight up-and-down movements of the air layers make the images change shape and apparent size. Even though it is mostly seen in polar latitudes, its Italian roots are owed to the fact that it was first described in the Strait of Messina, between continental Italy and Sicily. Figure 62 (right) shows our ray-traced image, using the temperature profile seen in Figure 63. Figure 64 shows our animation of the Fata Morgana effect.

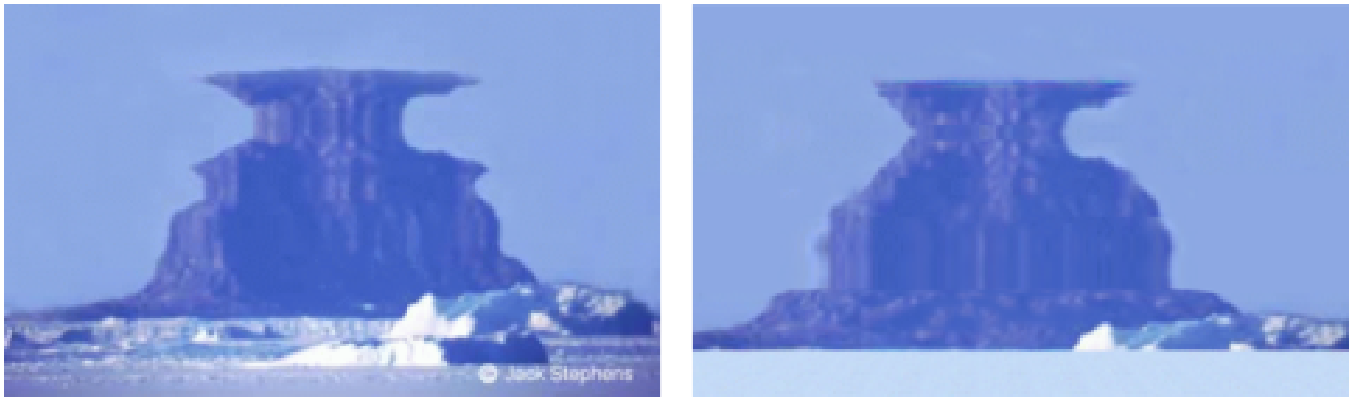


Figure 62: Left: Real picture of a Fata Morgana. Right: Rendered Fata Morgana.

15 Simulating Sunsets

We can also generate simulations of sunsets under certain distributions of the index of refraction of the atmosphere. We have chosen some of the most obvious and spectacular ones; again, some are well known whereas some are hard to see in a lifetime. The chosen effects are the flattened sun, the double sun, the split sun, the Novaya-Zemlya effect and the green flash. This last effect is specially interesting in the context of this course because it is closely tied to the mechanisms of human perception, as we will see.

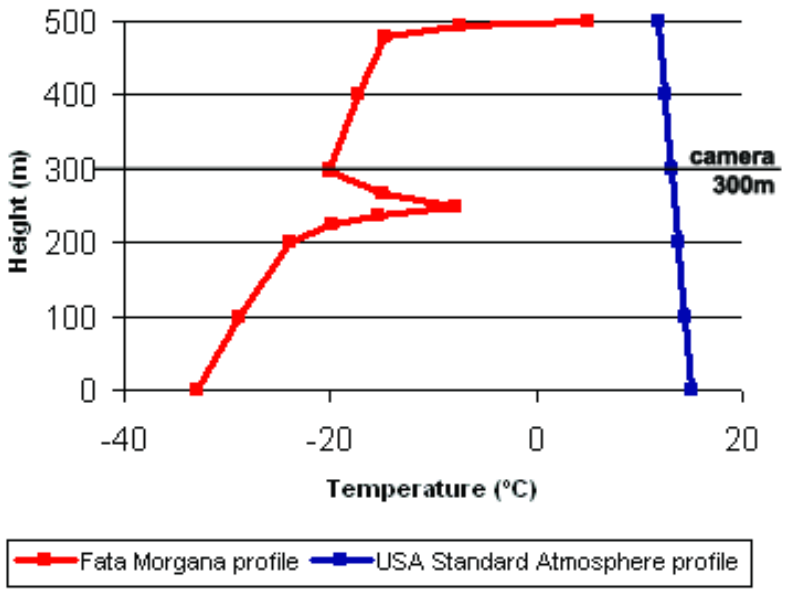


Figure 63: Temperature profile of the Fata Morgana rendered in figure 62 compared to the USA Standard Atmosphere's temperature profile.

Figure 64: Frames of an animation of the Fata Morgana.

For the simulations, the earth is represented as a sphere of 6375 length units of radius (once again, the units can be thought of as being kilometers). Another sphere of 695000 units of radius represents the sun. The distance between the spheres is 150 million units. The observer is located at a small distance above the surface of the earth. We have also added several image-based enhancements to the renders in the foreground to increase realism, but of course the sun shapes are all physically-based simulated.

The range of values chosen for the index of refraction match those found in the real atmosphere. This, along with the dimensions of the scene described above, which are also real-world values, assures the visual accuracy of the phenomena rendered.

15.1 Flattened Sun

This is the most common case that appears in an atmosphere in which the index of refraction decreases with height, because the density of the air decreases as one moves away from the earth, and so the index of refraction decreases as well. As a result of this, the sun is not seen as a perfect circle, but appears rather flattened along the vertical axis. This happens because the rays become curved downwards, towards the areas with a greater index of refraction (see figure 66). This causes a distortion of the image we receive from the sun. Moreover, the image of the sun appears above the area it should.

Figures 65 and 66 also show the distribution of the index of refraction for this scene, several frames of the results obtained and a real picture of the effect.

15.2 Double Sun

This is the phenomenon that takes place when there is a very thin layer of warm air over the surface of the earth. In this case, the rays traced from the observer that do not make contact with the ground are not affected by the warm layer and therefore the sun is perceived without distortion. However, the rays that do make contact with the warm layer become curved upwards. As a consequence, part of them intersects again with the sun, and as a result, we perceive a double image of the sun.

The effect is explained in Figure 68: it can be appreciated how the OA ray emitted from the observer O follows its normal path, with a slight downwards bending as explained in the flattened sun effect, until it reaches the sun. However, the lower OB ray arrives at the vicinity of the

earth, where the warm layer curves it upwards, then following a slightly curved path BC until it also reaches the sun. So hence we get two images: the normal one and the one generated by the rays that reach the warm layer near the earth that curves them upwards. The distribution of the index of refraction is also represented. To add realism, we have also included a decrease of the index of refraction as a function of height.

Figures 67 and 68 also show several frames of the resulting animation, plus a real picture of the effect. The difference between the effect generated through curved ray tracing and the real one is owed to the thickness of the warm layer and the position of the sun relative to the observer. Different combinations of these two parameters yield different outputs based on the same double-sun effect.

15.3 Split Sun

It is a curious phenomenon that may arise when there is a cold air layer near the earth and a warmer layer above it. The phenomenon of total reflection may take place when a ray passes from one medium to another with a smaller index of refraction with an adequate angle. In that situation there is no refracted ray, and the ray is reflected.

In figures 69 and 70 we have represented the earth and the layer of discontinuity, and we have traced three rays from the observer O. An important theorem we must consider when analyzing this phenomenon is that, of all the rays that pass through O, the horizontal is the one that forms the smallest angle with the circle, and so it is the one that forms the biggest angle with the normal to the surface of discontinuity. Hence, if the ratio of the indexes of refraction of both layers is big enough, the result will be that for an interval of directions above and below the horizontal, the ray emitted from the observer does not undergo refraction due to the phenomenon of total reflection, and so it is reflected downwards. This can be appreciated in the figure for the ray OB, and would also occur inside a small interval of directions around the direction OB. However, for the directions OA and OC, the incident angles are not so big and the ray is refracted.

In figure 70 we also show the distribution of the index of refraction used. This distribution causes the split sun effect, with two portions of the sun separated by an empty strip located around the horizontal plane that passes through the observer. Finally, several frames of the resulting simulation are included.

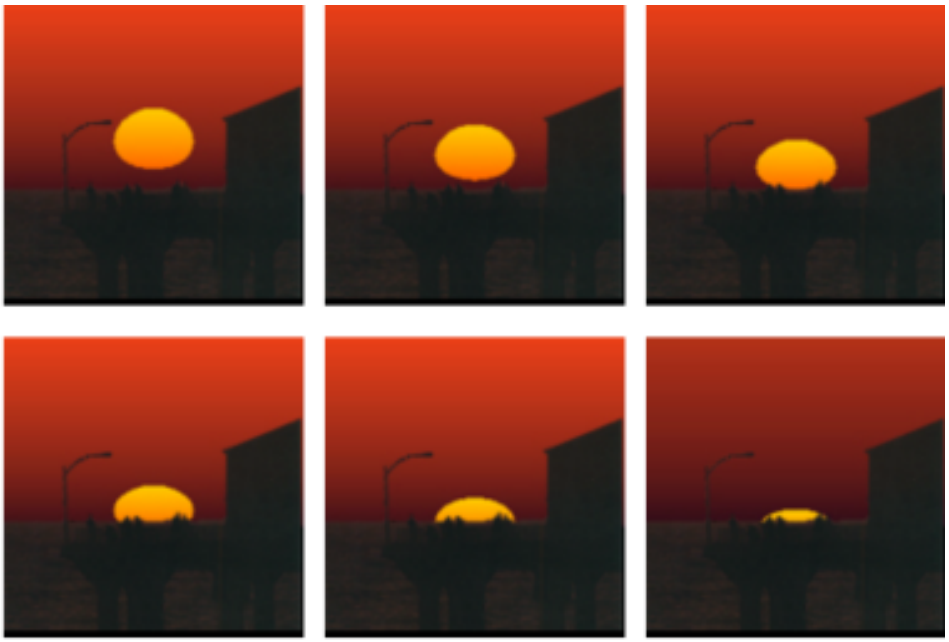


Figure 65: The flattened sun.

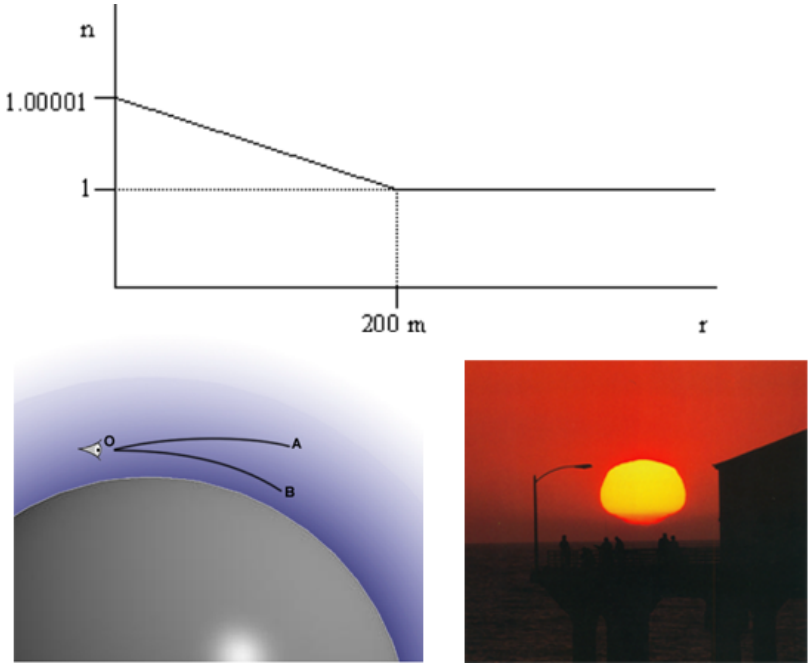


Figure 66: Index of refraction, light paths and real picture of the flattened sun

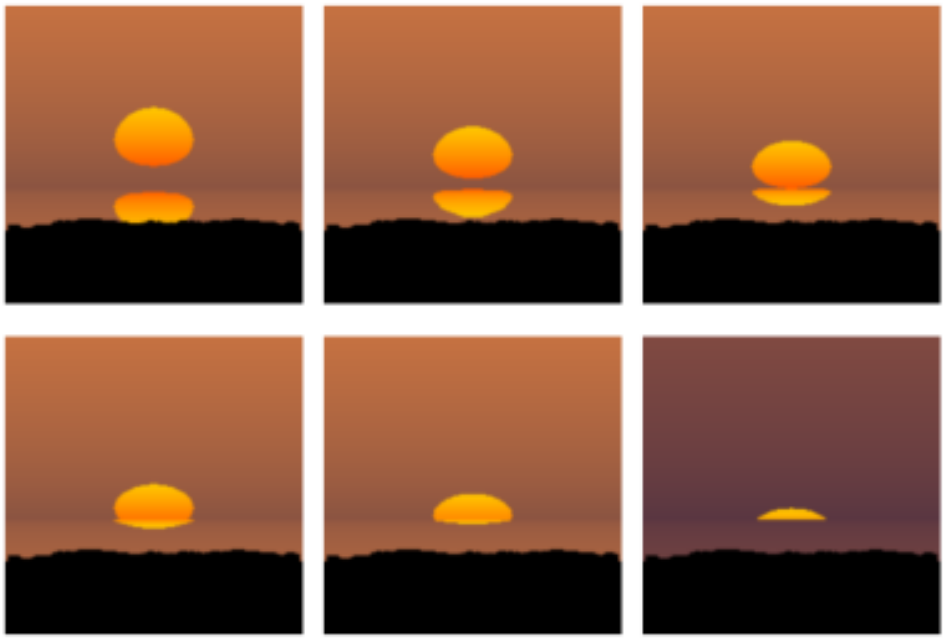


Figure 67: The double sun.

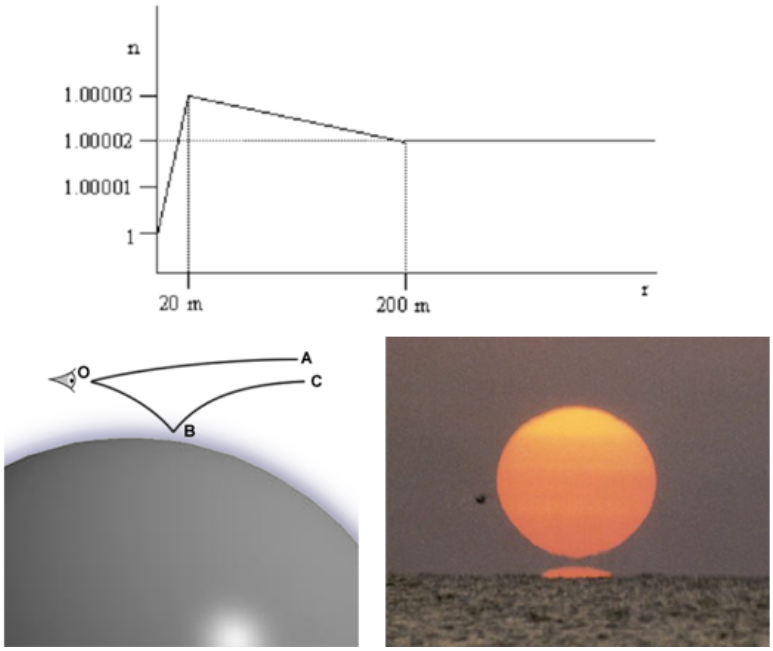


Figure 68: Index of refraction, light paths and real picture of the double sun

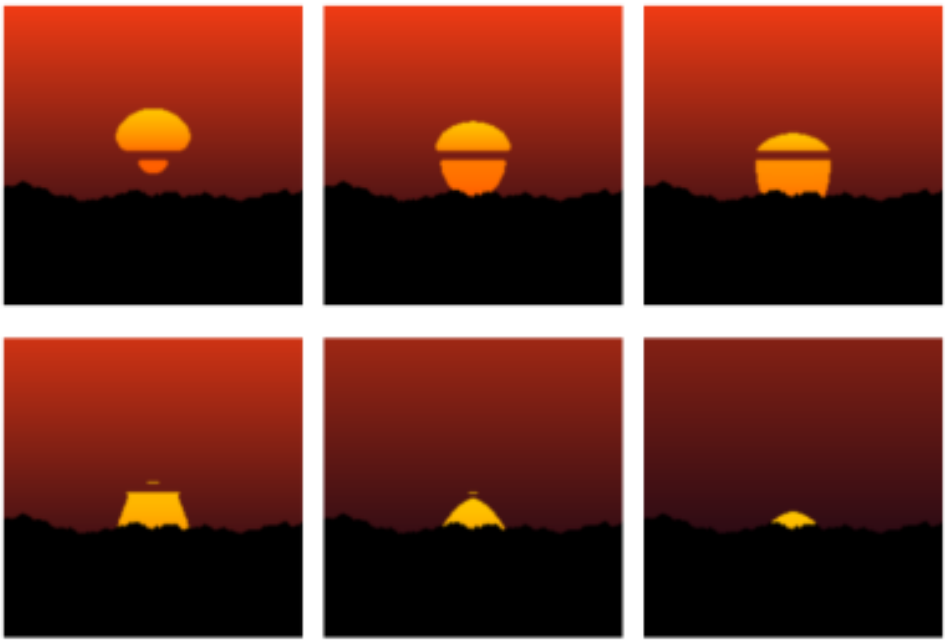


Figure 69: The split sun.

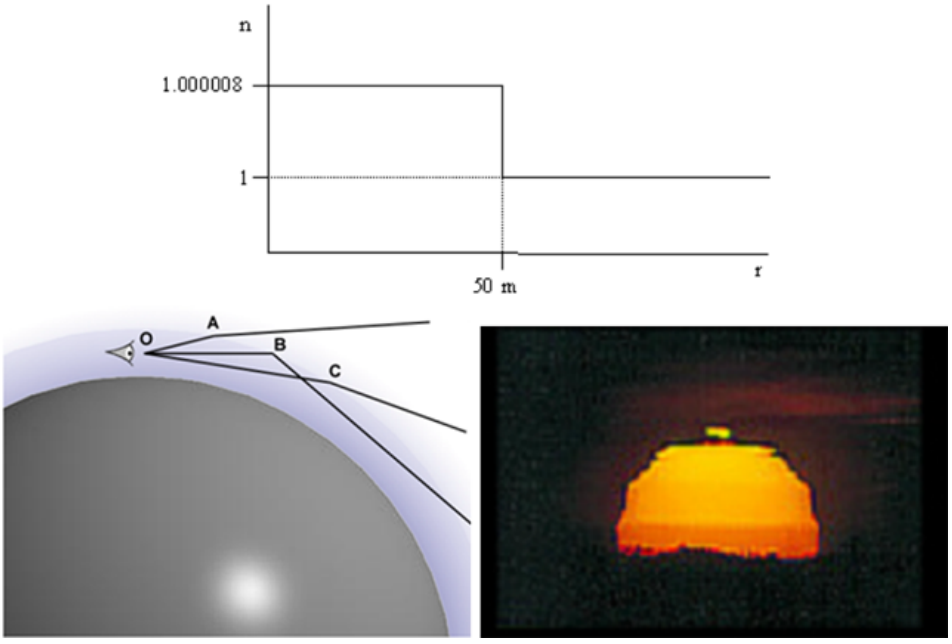


Figure 70: Index of refraction, light paths and real picture of the split sun

15.4 Novaya-Zemlya

In the winter of 1596, Dutch explorer Willem Barents and his crew were sailing off the north coast of the Russian arctic island of Novaya Zemlya. They were forced by thick arctic sea ice to spend the winter there, submerged in the long arctic night. One day in early March they were surprised to see a distorted sun appear for a short time above the horizon, almost two weeks ahead of time, bringing an early twilight to the island [Zee01].

What they saw was the Novaya-Zemlya mirage [vdWKL03], which describes a case when celestial objects, such as the sun, can be seen even when they are situated below the horizon; although it is developed by different reasons than the superior mirage. Steep arctic temperature inversions trap the sun-rays, making them travel within an inversion layer for hundreds of kilometers (a phenomenon known as *ducting*). If the layer has the right temperature gradient, the light bends following the curvature of the Earth over that long distance, effectively showing up above the horizon, even though the sun might really be up to five degrees below it.

With long-term scientific polar settlements established over the past few decades, this phenomenon has been widely photographed. Once the sun has burst above the horizon, its light still remains trapped within the inversion layer, distorting it into an unusual rectangular shape. Figure 71 shows a real Novaya-Zemlya effect, alongside our ray-traced image. Figure 73 shows our animation of the Novaya Zemlya effect.

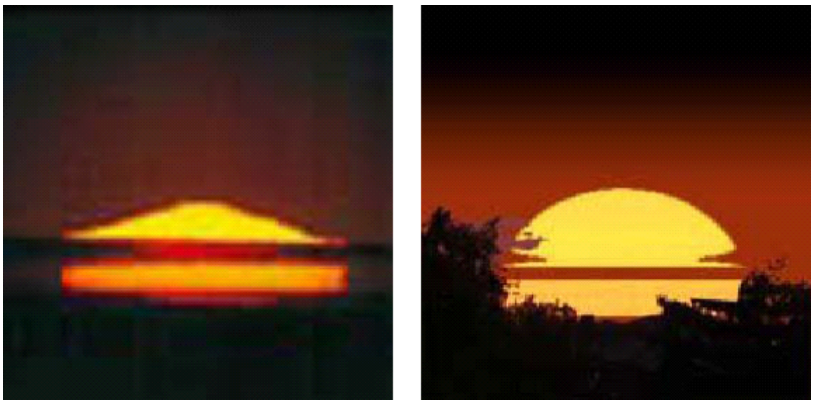


Figure 71: Left: Real picture of a Novaya-Zemlya effect. Right: Rendered Novaya Zemlya.

The atmospheric profile can be seen in 72. In order to get a better view of the effect, the camera has been placed inside the inversion layer, two meters above ground level.

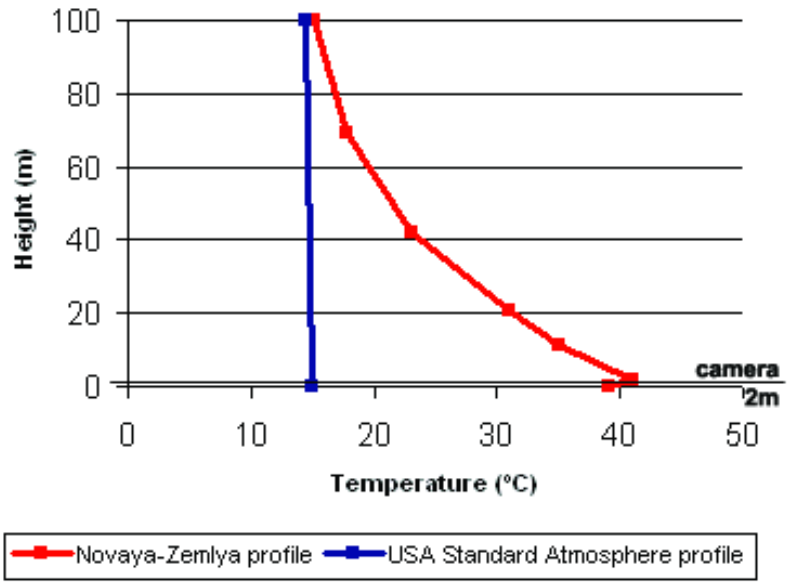


Figure 72: Temperature profile of the Novaya Zemlya effect rendered in Figure 71 compared to the USA Standard Atmosphere's temperature profile.

Figure 73: Frames of an animation of the Novaya Zemlya.

15.5 The Green Flash

The green flash [You] is a real phenomenon (as opposed to an optical illusion as it is said in some texts), which occurs under very specific conditions both during sunrise and sunset. A small part of the sun will change its color, from red-orange to blue-green. lasting for about two seconds[5].

The cause of this effect is the dispersion of light as it traverses the atmosphere: the beam of white light gets split into all the different wavelengths. Since the index of refraction is a function of wavelength, the net result is that the paths of light are curved depending on the wavelength (Figure 74). This dependence is so weak that most of the time remains unseen, hidden by Rayleigh or MIE scattering.

Figure 74: Dispersion of light while traversing an inhomogeneous medium.

The explanation of the green flash effect is pretty straightforward once dispersion is taken into consideration: the closer the sun is to the horizon, the longer the light paths from the sun to the observer. Since the atmosphere is denser at low heights (its index of refraction is higher), light rays get bent downwards, and because of its wavelength dependency, blue and green rays are bent more than orange-red ones. The result is a green-blue halo on top of the solar disc, and a red rim under it (see Figure 75 for a visual explanation). Aerosols in the atmosphere and Rayleigh's dispersion [Irw96] eliminate low wavelengths (blue) so what remains is the green that gives the name to the green flash. A picture of the phenomenon can be seen in Figure 76, left. The red rim of the green flash is usually invisible under the horizon, although it can be seen sometimes as in Figure 76, right.

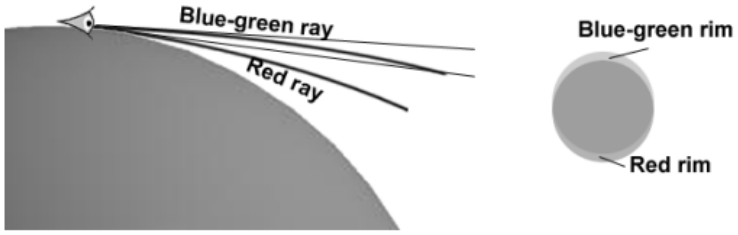


Figure 75: Explanation of how atmospheric dispersion generates a green flash.

It is also true that under even more special atmospheric circumstances, others colors can be seen, as in Figure 77. These colored flashes are due to the same atmospheric phenomena than the green flash itself. They are due also to Rayleigh's dispersion. Sometimes this dispersion is not strong enough and still some low wavelengths remain, so the flash can be blue or cyan. On the other hand, given the adequate circumstances, even medium wavelengths in the greenish range can be dispersed, the result being a yellow flash.

[5] Jules Verne [Ver82] described the green flash as *a green which no artist could ever obtain on his palette, a green of which neither the varied tints of vegetation nor the shades of the most limpid sea could ever produce the like! If there is a green in Paradise, it cannot be but of this shade, which most surely is*

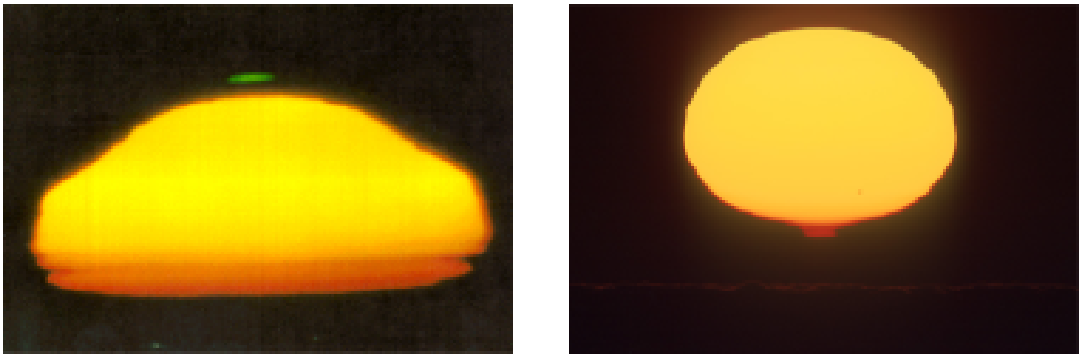


Figure 76: Left: The green flash. Right: The red flash.

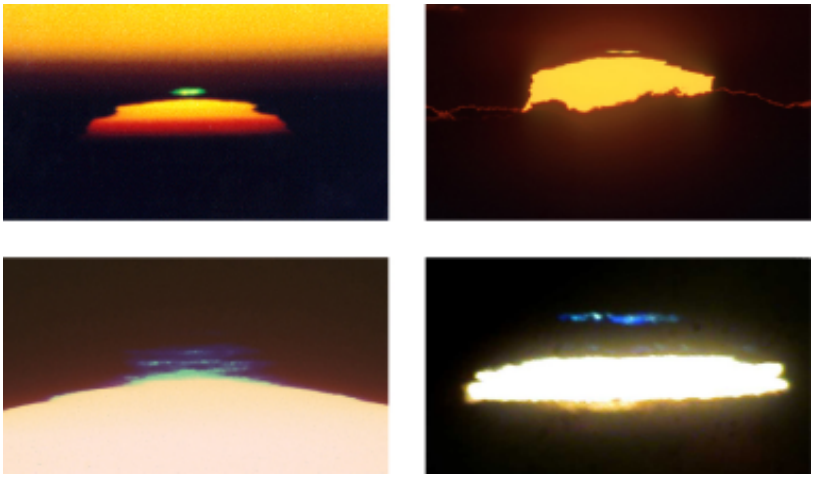


Figure 77: Different colors of the green flash: green, yellow, cyan and blue.

The green rim in the solar disc is too small for the naked eye to see, but its effect is usually magnified by some kind of mirage. Figure 78 shows our simulation of the green flash. The atmospheric profile that has generated the green flash is shown in Figure 79. Also Rayleigh's scattering and bleaching (see next section) have been considered.

Figure 78: The green flash. Left: real picture; right: simulation.

the true green of Hope.

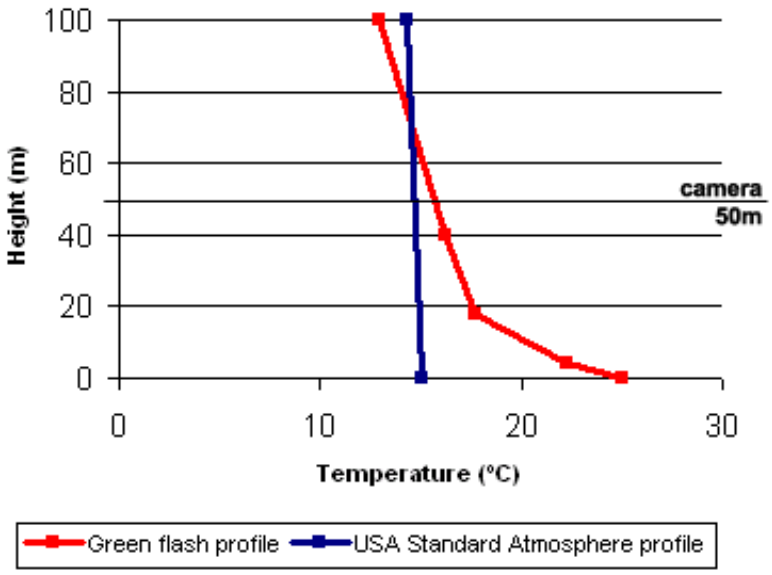


Figure 79: Temperature profile of the green flash effect rendered in figure 78 compared to the USA Standard Atmosphere's temperature profile.

15.5.1 Bleaching

The green flash is specially interesting since the effect is not only caused by the dispersion of the atmosphere (as presented before), but it is also magnified by our visual system. The retina of the eye is a compound of photo-receptors called rods and cones. Rods are very sensitive under low lighting conditions whereas cones are active in photopic luminance levels. According to the part of the visible spectrum to which they are most sensitive, the cones can be referred to as long-, middle- and short-wavelength-sensitive (L, M and S). When the light bleaches the pigment in a photoreceptor its density decreases. The loss of concentration in the photo-pigment makes the absorption of light diminish and the range of spectral sensitivity becomes narrower around the wavelength peaks. In cones, the effects of bleaching on pigment concentration can have large effects on spectral sensitivity.

When waiting for the green flash to occur, the retina has been exposed to very bright red light for relatively a long time. That bleaches some of the red-sensitive photo-pigments in the L cones. This degeneration has a great effect on our spectral sensitivity, and the now less sensitive cones will perceive red as yellow, and yellow as green. Consequently the green rim of the green flash will become even greener, as it can be seen in our tests simulations in Figure 80. A more complete description of the bleaching model used can be found in [GAMS05].

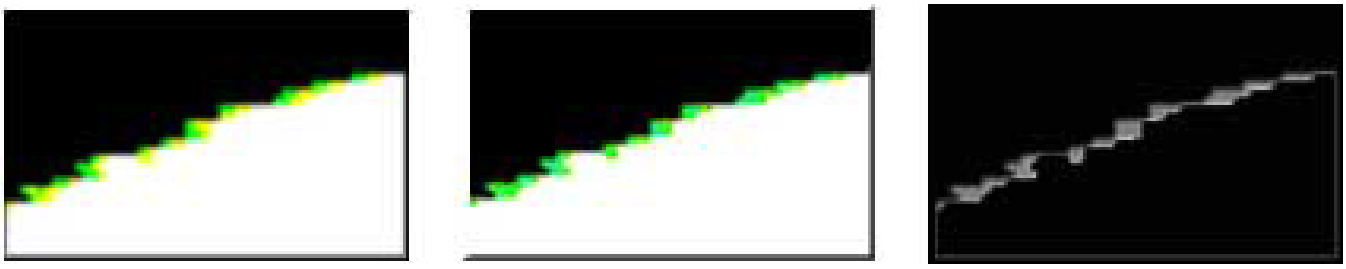


Figure 80: Left: a simple test simulation of the green rim of a green flash. Middle: the same simulation after the application of the bleaching algorithm. Right: Green pixels enhanced by bleaching

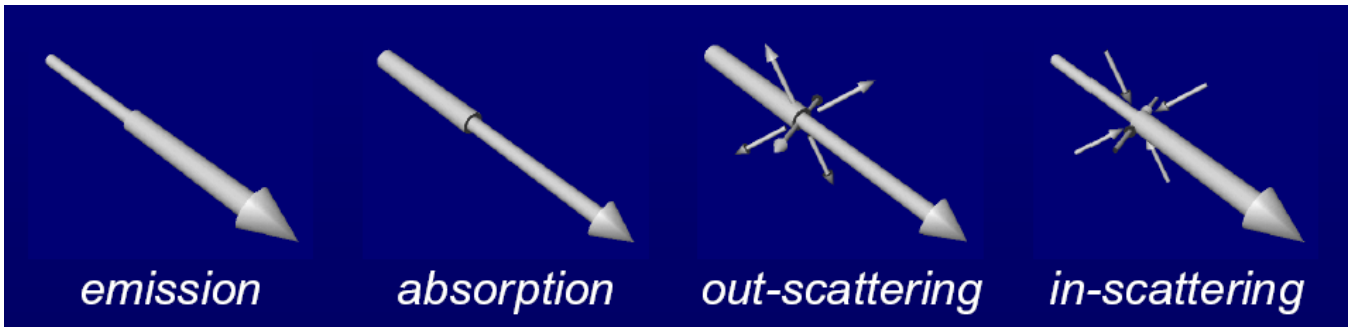


Figure 81: Radiance variation in a participating medium: absorption, emission, in-scattering and out-scattering [PPS97].

16 Participating Media

Unless you are a creature from outer space, chances are you, like me, live immersed in Earth's atmosphere (which suits us just fine come breathing time). The atmosphere contains in fact particles, aerosols, dust and light traveling through it interacts with them at every single differential step. The colors of the sky, the loss of visibility in a foggy day, the so-called aerial perspective (loss of color perception with distance)... are all effects of this interaction of light with the invisible particles of the atmosphere. The type of media where these interactions occur is called *participating media*.

Sometimes (probably *most* of the times) the influence of participating media can be ignored while producing an image, thus simplifying the rendering. But several phenomena cannot be simulated without taking them into account, and they might be crucial for a realistic, high-fidelity depiction of the scene being simulated. In fact, using one of the examples given above, aerial perspective is one of the most important depth cues in outdoors scenes; whilst one can always come up with tricks to mimic it, we are interested in this section in a physically-based approach which will provide us with a better understanding of the phenomena involved.

A very thick participating medium, where lots of interesting effects take place, is water; we will in fact concentrate on underwater imagery for this section, although the explanations offered are general an can be applied to any other participating medium. Clouds are another example of a rich participating medium; the reader can refer to [DEF*04] for an in-depth treatment of several different approaches to rendering clouds, both realistic and/or interactive.

16.1 The Physics of Participating Media

Four types of events can contribute to the radiance variation in a participating medium: absorption, emission, in-scattering and out-scattering (see figure 81). Usually, participating media algorithms solve the integro-differential Radiative Transfer Equation (RTE), which takes into account emission, absorption and elastic scattering, but does not yield a solution for inelastic scattering events.

The following derivation of the RTE peruses the work of Wann Jensen [JC98] and is not meant to be exhaustive. In its wavelength-dependent form, the RTE can be formulated as:

$$\frac{\partial L_\lambda(x,\vec{w})}{\partial x} = \alpha_\lambda(x)L_{e,\lambda}(x,\vec{w}) + \sigma_\lambda(x)L_{i,\lambda}(x,\vec{w}) - \alpha_\lambda(x)L_\lambda(x,\vec{w}) - \sigma_\lambda(x)L_\lambda(x,\vec{w}) \tag{19}$$

where $\frac{\partial L(x,\vec{w})}{\partial_x}$ represents the variation of radiance L at a point x in the direction $\vec{w}$, α and σ are the absorption and scattering coefficients, L_e is the emitted radiance and L_i is the in-scattered radiance. Defining the extinction coefficient as $\kappa_\lambda(x) = \alpha_\lambda(x) + \sigma_\lambda(x)$ and integrating $L_{i,\lambda}$ over the sphere Ω we get:

$$\frac{\partial L_\lambda(x,\vec{w})}{\partial x} = \alpha_\lambda(x)L_{e,\lambda}(x,\vec{w}) + \sigma_\lambda(x)\int_\Omega p_\lambda(x,\vec{w}',\vec{w})L_\lambda(x,\vec{w}')d\vec{w}' - \kappa_\lambda(x)L_\lambda(x,\vec{w}) \tag{20}$$

which is the integro-differential, wavelength-dependent RTE governing the transport of light in participating media. However, this equation does not account for energy transfers between wavelengths, the phenomena known as inelastic scattering.

Inelastic scattering implies an energy transfer from wavelength λ' to λ, with $\lambda' < \lambda$ within the visible spectrum, and gives rise to fluorescence and phosphorescence phenomena. Fluorescence occurs when a molecule absorbs a photon of wavelength λ' (called excitation wavelength),

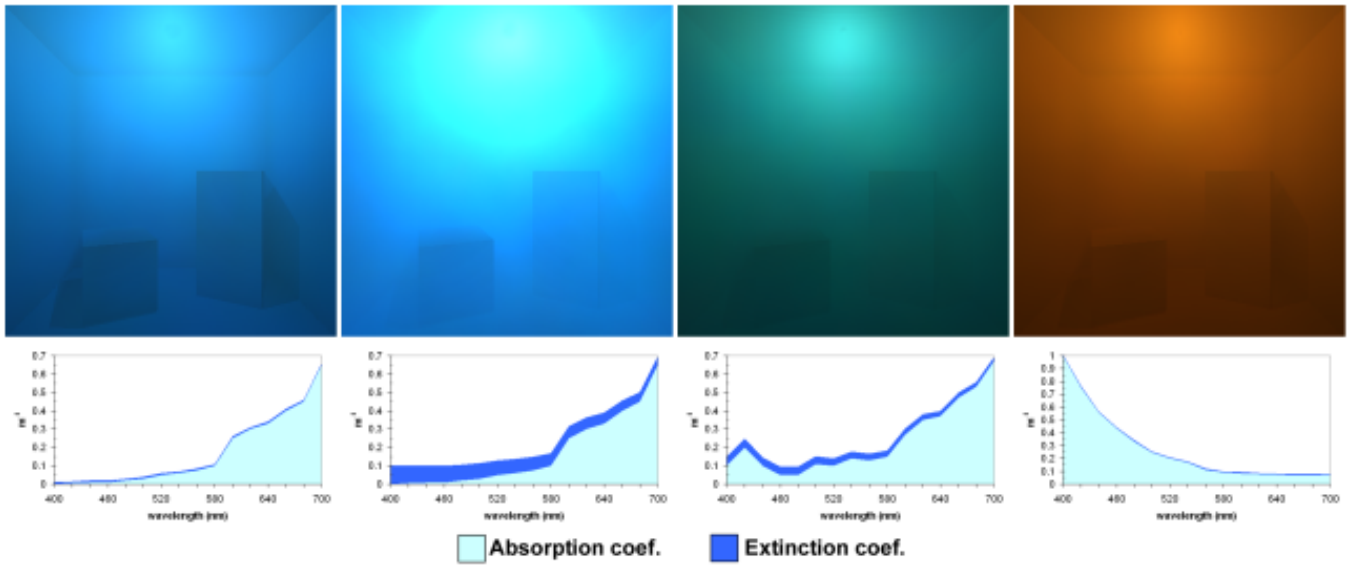


Figure 82: Fluorescent ocean water in Cornell rooms. (a), (b) and (c) show varying concentrations of chlorophyll ($0.05mg/m^3$, $0.1mg/m^3$ and $5mg/m^3$ respectively). (d) High concentration of yellow matter ($5mg/m^3$).

and re-emits it at a longer wavelength λ according to a *fluorescence efficiency function* $P_f(\lambda)$. The time lapse between the two events is 10^{-11} a 10^{-8} seconds, so for Computer Graphics it can be taken as an instantaneous process. For pure substances, re-emission is isotropic and the wavelength of the re-emitted photons is independent of the different excitation wavelengths, although the intensity of the re-emission does depend on them. Phosphorescence is a similar process, governed by the *phosphorescence efficiency function*, with the main difference being that the re-emitted energy declines with time according to the function $d(t)$.

To be able to compute these inelastic scattering events, we need to develop the RTE equation further, by adding a term that accounts for such energy transfers. This term can be expressed as a double integral over the domains of the solid angle and wavelength:

$$\int_{\Omega}\int_{\lambda} \alpha_{\lambda_i}(x) f(x,\lambda_i \rightarrow \lambda) L_{\lambda_i}(x,\vec{w}') \frac{p_{\lambda}(x,\vec{w}'_i,w)}{4\pi} d\vec{w}_i d\lambda_i \tag{21}$$

where α_{λ_i} is the absorption coefficient for wavelength λ_i (there is no inelastic scattering without previous absorption), $f(x,\lambda_i \rightarrow \lambda)$ is the function that governs the efficiency of the energy transfer between wavelengths, defined as the probability of a photon of λ_i being re-emitted at λ, and $p_{\lambda}(x,\vec{w}'_i,w)$ is the phase function that defines the re-emission direction. For fluorescence and phosphorescence, this phase function is isotropic [Mob94]. Adding this term to the RTE (equation 20) we obtain the Full Radiative Transfer Equation (FRTE) [Gla95a]:

$$\begin{aligned} \frac{\partial L_{\lambda}(x,\vec{w})}{\partial x} &= \alpha_{\lambda}(x) L_{e,\lambda}(x,\vec{w}) + \\ \sigma_{\lambda}(x) \int_{\Omega} p_{\lambda}(x,\vec{w}',\vec{w}) L_{\lambda}(x,\vec{w}') d\vec{w}' &- -\kappa_{\lambda}(x) L_{\lambda}(x,\vec{w}) + \\ \int_{\Omega}\int_{\lambda} \alpha_{\lambda_i}(x) f(x,\lambda_i \rightarrow \lambda) L_{\lambda_i}(x,\vec{w}') \frac{p_{\lambda}(x,\vec{w}'_i,w)}{4\pi} d\vec{w}_i d\lambda_i & \end{aligned} \tag{22}$$

which is the equation that must be solved to take into account multiple inelastic scattering in participating media, thus being able to render volume fluorescence effects.

A full description of the implementation of the FRTE in a global illumination renderer has been previously published in [GMAS05].

16.2 Underwater Environments

Underwater virtual environments can be depicted in a high-fidelity way following the approach described in the previous sections. Different events occur during the interaction of light and water that govern its final color. This color will depend on the light sources (the sun, the sky, any artificial source inside or outside the water body) and on the absorption and scattering events. A realistic depiction of these events require that multiple, anisotropic scattering be modeled accurately at the spectrum level, which is computationally expensive and thus most works usually assume strong simplifications (such as calculating only single scattering).

Pure seawater absorbs most wavelengths except for blue: the absorption coefficient peaks at 760 nanometers, and reaches a minimum at 430 nm. Scattering β is modeled as the scattering in pure sea water plus the scattering caused by the suspended particles, as proposed in [Mob94] ($\beta = \beta_w + \beta_p$). For pure water we use a phase function similar to Rayleigh's:

$$\beta_w(\Psi) = 0.06225(1 + 0.835cos^2\Psi) \tag{23}$$

while the scattering caused by particles is modeled using a Henyey-Greenstein phase function with $g = 0.924$:

$$\beta_p(\Psi, g) = \frac{1-g^2}{(1+g^2-2gcos\Psi)^{3/2}} \tag{24}$$

It is very common in ocean waters to see a color shift ranging from greenish to very bright green, or even yellowish. These hue shifts are owed to the variation in the concentration and type of the suspended microorganisms, mainly phytoplankton, which presents a maximum absorption at 350 nm. rapidly decreasing to almost zero beyond 500 nm. The most important element in the phytoplankton is chlorophyll, which presents spectral absorption peaks in the blue and red ends of the spectrum and is the most important source of volume fluorescence in the waters. For chlorophyll, $\Gamma^c(\lambda_i)$ is wavelength-independent, with values ranging from 0.01 to 0.1 (we use the superscript c for chlorophyll). As with most inelastic scattering event, the re-emission phase function is isotropic.

Another important source of fluorescence is the Color Dissolved Organic Matter (CDOM), also called yellow matter, present in shallow ocean waters and harbors. $\Gamma^y(\lambda_i)$ is also wavelength-independent, with values between 0.005 and 0.025, and re-emission is also isotropic. Appendix B of this chapter presents a more detailed description of the functions describing fluorescent phenomena owed to chlorophyll and yellow matter.

For comparison, Figure 82 shows different colorations of ocean water, according to varying chlorophyll and yellow matter concentrations which trigger inelastic scattering events with different probabilities. Below each picture, the resulting absorption and extinction curves (functions of the different concentrations of chlorophyll in the virtual waters) are shown for each case. Image (a) shows little fluorescence (low chlorophyll concentration of $0.05mg/m^3$), and the waters are relatively clear. When chlorophyll concentration increases, fluorescence events become more prominent and the image first gets a milky aspect (b), losing visibility and reaching a characteristic green hue when chlorophyll reaches $5mg/m^3$. Image (d) shows fluorescence owed to yellow matter. The absorption function in this case has been modeled after [Mob94]: $a_y(\lambda) = a_y(440)^{-0.014(\lambda-440)}$ where $a_y(440)$ is the empirical absorption at 440 nm.

Figure 83 shows the importance of modeling underwater environments based on the laws of physics, in order to provide an accurate depiction. It can then be used as a predictive tool for simulations in fields so diverse as underwater archeology, planning rescue missions, laying submarine cables...). All of the images are lit by a Philips SW-type© luminaire, specified according to the CIBSE TM14 format (which is also an important feature in predictable virtual scenarios).

Fluorescence owed to inelastic scattering is computed according to the varying concentrations of chlorophyll in each image (between 0.01 and $0.1mg/m^3$). The top two images represent a sunken boat along a Happy Buddha in clear, shallow waters (left) or deep underwater with a chlorophyll concentration of $0.05mg/m^3$(right). For the bottom-left image, we have added a volume temperature field that simulates a heat source outside the image as explained in [SGGE05], deriving the index of refraction using the formula $n = 1 + \frac{T_o}{T}(n_o - 1)$ as proposed by Stam and Languenou [SL96]. The distortions caused by the varying index of refraction are visible in the image. The bottom-middle image uses a smoke-like medium, modeled as a 3D turbulence function, whereas the last to the right shows the effects of a highly anisotropic medium.

17 An Industrial Perspective

Rendering physically-based imagery by using global illumination techniques have become very useful and necessary in certain areas of interest like safety, military or industry. The applications in these areas usually deals with the analysis of visual perception under certain unfavorable environmental conditions, where the presence of a medium has a noteworthy influence in the visibility (and therefore the design) of certain elements such as road signs, fire exit signs, car headlamps, road lighting, etc. Examples of these participating media include smoke, fog, dust, flames, silty, and abyssal waters or atmospheric phenomena. As we have seen, simulating these participating media implies the correct computation of the absorption and scattering of light and therefore it has been always computationally expensive.

Nevertheless, *realistic*, *as closely as possible*, *high fidelity visual scenes*, *detail and realism*... these are all words taken from the web sites of different companies offering some kind of service regarding virtual environments. So the drive to achieve *there-reality* (as real as being there, term coined by Alan Chalmers) exists. Companies strive to offer that extra level of realism, to go from game-like graphics to is-this-real? graphics.

Some of the most popular fields of application include aviation (both civil and military), oil, driving simulators (trains, buses, cars, trucks...), nuclear power-plants and, the entertainment industry or the military in general. Driving simulators, for instance, need to be able to simulate a wide range of weather conditions: sunny, falling snow, foggy, ice on the road... as well different times of the day: night/day/dusk. Participating media and effects such as subsurface scattering need to be taken into account for a high-fidelity depiction of the scene (see figure 84). The human brain extracts a lot of information from an image, and certain visual cues trigger additional data, such as the perception of distance from aerial perspective, heat from a mirage in the road or cold from seeing one's breath.

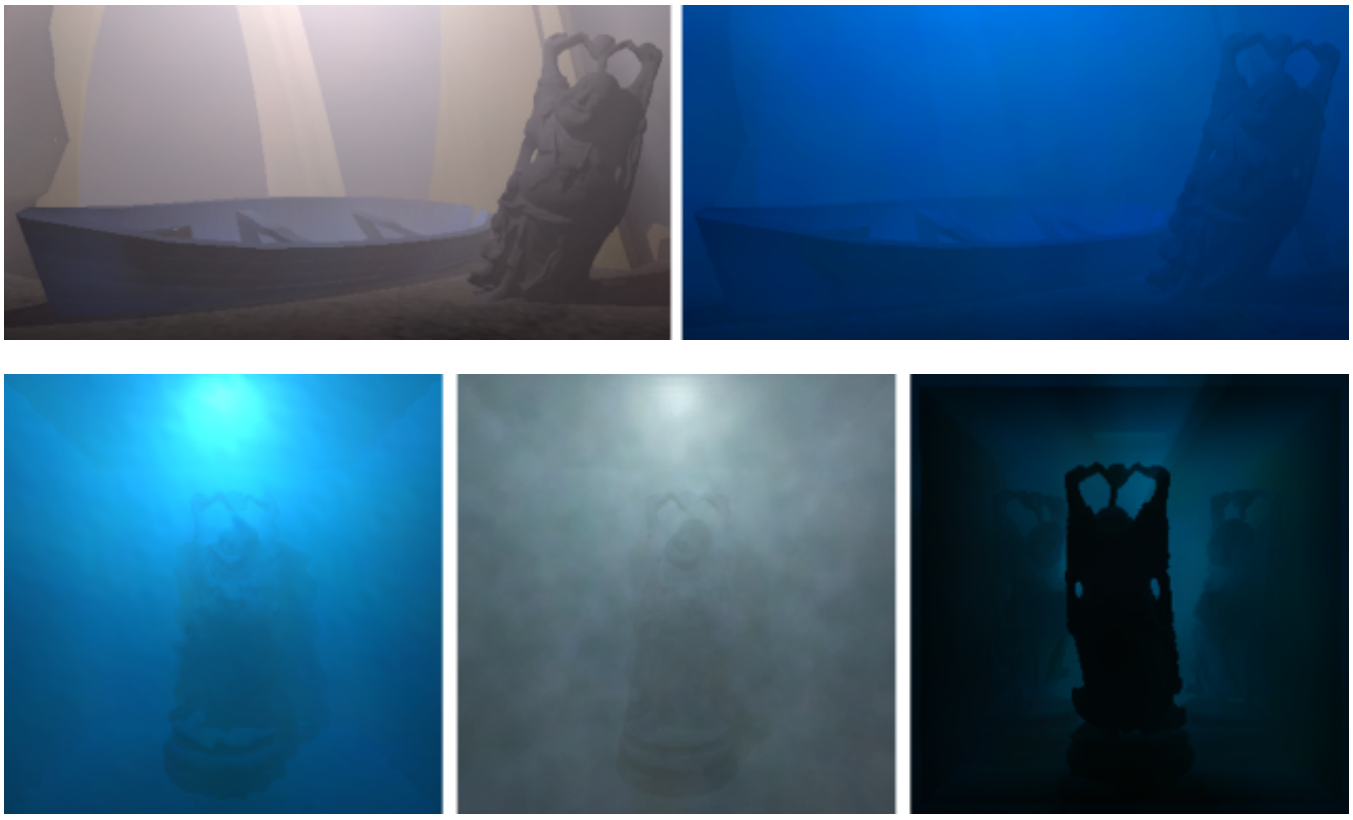


Figure 83: Different images with inelastic scattering in participating media. Top left: very low chlorophyll concentration. Top right: higher concentration yields more inelastic scattering events. Bottom left: distortions caused by a 3D temperature field. Bottom middle: 3D turbulence field simulating smoke. Bottom right: highly anisotropic medium.

Figure 85 shows several snapshots taken from CAE (www.cae.com), a provider of simulation and modeling technologies and integrated training services for civil aviation, amongst other things[6]. Effects such as aerial perspective, participating media (clouds), MIE and Rayleigh scattering have been simulated to achieve greater realism. Whereas the simulations shown in the images are just real-time hacks, more computing power (in any form: sheer CPU velocity, GPU's, PPU's (Physics Processing Units), parallelism...) plus clever algorithms should give us real-time, physically-based simulations in a future not so far away. Similarly, accurate underwater simulations can be used in rescue missions, archaeological research or cable tracking...

17.1 The Need for Physical Accuracy

We could try and define the term *simulation* as a procedure that allows us to study a physical system by substituting it for another one which is more suitable for observation or measuring. Different interests can be addressed by means of simulation. An astronaut can learn to pilot a space shuttle, a government can predict the evolution of the population in its country, a meteorologist can predict storms or a car company can preview its brand new model before it is actually built. These different approaches require different layers of complexity in the simulation: the astronaut trainer, for instance, will most likely need to be able to simulate the dynamics of the system to a certain degree. On the other hand, the car company might not need such realism if only the visual appearance is to be discussed by means of the simulation. But once the general shape is approved, it might then need to undergo a simulated wind tunnel test, which will require a new layer of complexity in the simulation. Thus, motivation guides the simulation itself.

Several simulations will work better by using Virtual Reality techniques. These techniques create immersive environments that make the interaction between man and the simulation as indistinguishable from reality as possible. Most of the time this implies the concept of real time, although its real necessity will be driven by the goal of the simulation.

Different applications greatly differ in requirements for data accuracy and visualization quality. Even using state-of-the-art hardware and algorithms, it is not possible for a Virtual Reality system to support all the different visualization demands at the same time. A user might just want to see detailed shape information for a given object, with accurate illumination or final material appearance being secondary. In that case, the information can most likely be presented based on real-time graphics. Other times the latter might be precisely what the user needs, and rendering accurate physically-based illumination on the fly is a task for which real time is not prepared, at least not yet and without any precomputations. This balance between real-time and render quality is one of the main issues when designing virtual environments.

[6]This company has been selected as an example for no particular reasons, and the author has no relation with it whatsoever

Figure 84: Real picture of a road in a foggy night. Note the strong scattering effect caused by the participating medium around the light sources, as well as the lack of clear visibility.

As we have seen, precise calculations of light paths and energy transfer in inhomogeneous, participating media comes at a cost. It is not cheap, and one can ask oneself *Is this worth it?* Do we need to be this precise, so physically-based? Well, as we have seen before, the motivation of the simulation should guide the precision behind it. You might not need all the bells and whistles for a shoot'em-up computer game, where other aspects might be considered more important. But even so, recent games such as *Half Life II* are incorporating physics engines that crank up an extra notch or two of realism.

Where is the limit? Well, we could argue that human perception should be the ultimate standard to define the limits of our simulation. In other words, if it's not going to be perceived by the human observer, why bother? On the other hand, it can also be argued that we could use simulations to gain knowledge *beyond* the limitations of human perception: it would be nice to query the simulation about, say, the exact luminance of a specific surface or the angle of incidence of a given ray (something our perception mechanisms are not very good at). One thing is true, though: the only reason why we are forced to cut corners and settle for imperfect simulations is computing power. We wouldn't have come up with so many tricks and hacks if we could (magically) have instant access to infinite computing power. With the current evolution in affordable graphics hardware, GPU's and the new PPU's, it makes sense to keep pushing the limits of how real our simulations can get.

17.2 Case Study: the Temple of Kalabsha

For a practical study of how participating media and perceptual issues affect the depiction of a scene, we have chosen an archaeological setup, more specifically the Egyptian Temple of Kalabsha. The temple was dedicated to the Nubian fertility and solar deity known as Mandulis and the walls are covered with text and inscriptions depicting Egyptian deities such as Isis and Osiris. A more complete description of the temple can be found in [SV04][7].

The temple was originally built at Kalabsha (Talmis). However, the construction of the Aswan High Dam in 1959 forced it to be dismantled and moved to a new site. It is well known that the ancient Egyptians worshiped the sun; however, the new location and orientation of Kalabsha meant that it would now be impossible to visualize the effect of the sun on the temple, as it would have appeared to them. A virtual reconstruction was then created (see figure 86).

[7] Special thanks to the authors of [SV04] for the background information on the temple

Figure 85: Rendering natural phenomena in today's industrial applications (www.cae.com).

Figure 86: A real picture and the virtual reconstruction of the temple of Kalabsha (image on the right by Veronica Sundstedt and Patrick Ledda).

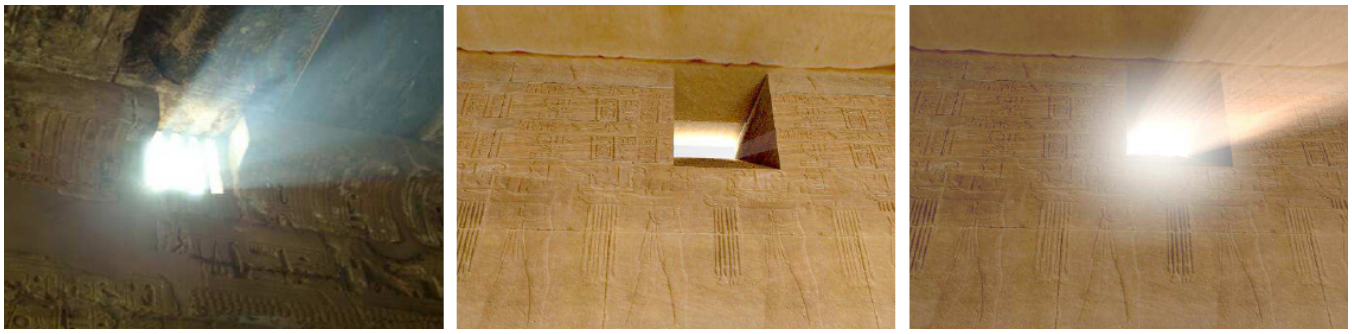


Figure 87: Left: real picture of light scattering after entering through a narrow slit. Middle: Simulation inside the Kalabsha temple, without participating media. Right: Simulation inside the Kalabsha temple, with participating media.

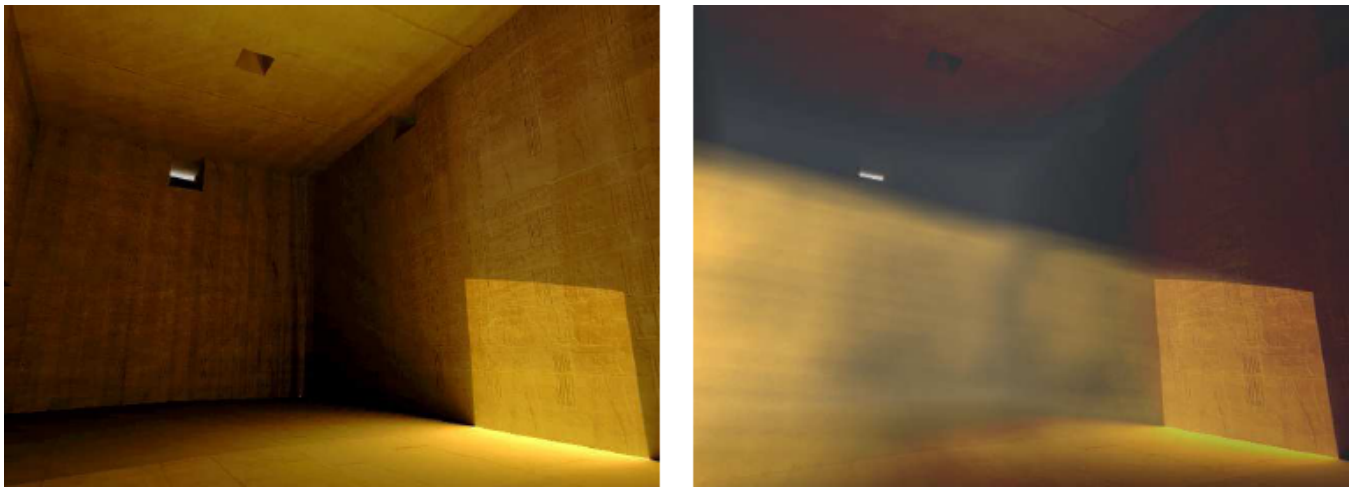


Figure 88: Another view of an inner chamber of the Kalabsha temple, without (left) and with (right) participating media included in the simulation.

But to accurately recreate how the sun-rays looked inside the temple, we also need to incorporate participating media in the simulations, in this case in the form of dust. The perception of the interior changes dramatically when the results of multiple scattering are added, and it more closely mimics the look it might have had two millennia ago.

The inner chambers of the temple had narrow slits to allow sunlight in. These slits were in fact so narrow that there is very little direct light contribution from there, whereas the participating medium (dust and sand) plays a key role in the transport of light throughout the scene. Figure 87 shows a close-up shot of a narrow slit from another site with similar architecture, for reference, along with our simulations inside Kalabsha without and with participating media taken into consideration. Sunlight scattering greatly alters the perception of the scene, and including it in the simulation mimics reality in a more accurate way. Figure 88 shows another simulation, this time with light entering through a door. Finally, figure 89 shows some frames from an animation of sunlight through the slit. For these, a tone mapping operator which includes a model of human perception was used, based on [LRP97]. A more complete description of the worked carried out for the Kalabsha simulation can be found in [SGGC05].

18 Open Challenges

It is well known that our visual system changes substantially under scotopic (below $0.034 cd/m^2$, photopic (above $3.4 cd/m^2$ or mesopic (in between) conditions, so a system aiming at delivering high-fidelity graphics must incorporate these changes; just solving the interaction between light and matter will not cut it anymore, the observer behind the phenomena needs also to be included in the simulation!

Coming up with a complete model of the human visual system is a multidisciplinary task which demands efforts from very different areas. We computer scientists need to build on previous studies by physiologists, opticians, physicists... The inherent subjective character of human perception is, by nature, very difficult to grasp, and thus some results and conclusions need to be based in a statistical analysis of multiple tests with human observers.

Two are the main open challenges: first, we need an *über-model* of human perception; our weak understanding of how the eye and the brain interact forces us to come up with incomplete, separate models that seem to work fine under certain circumstances or for a specific effect, but

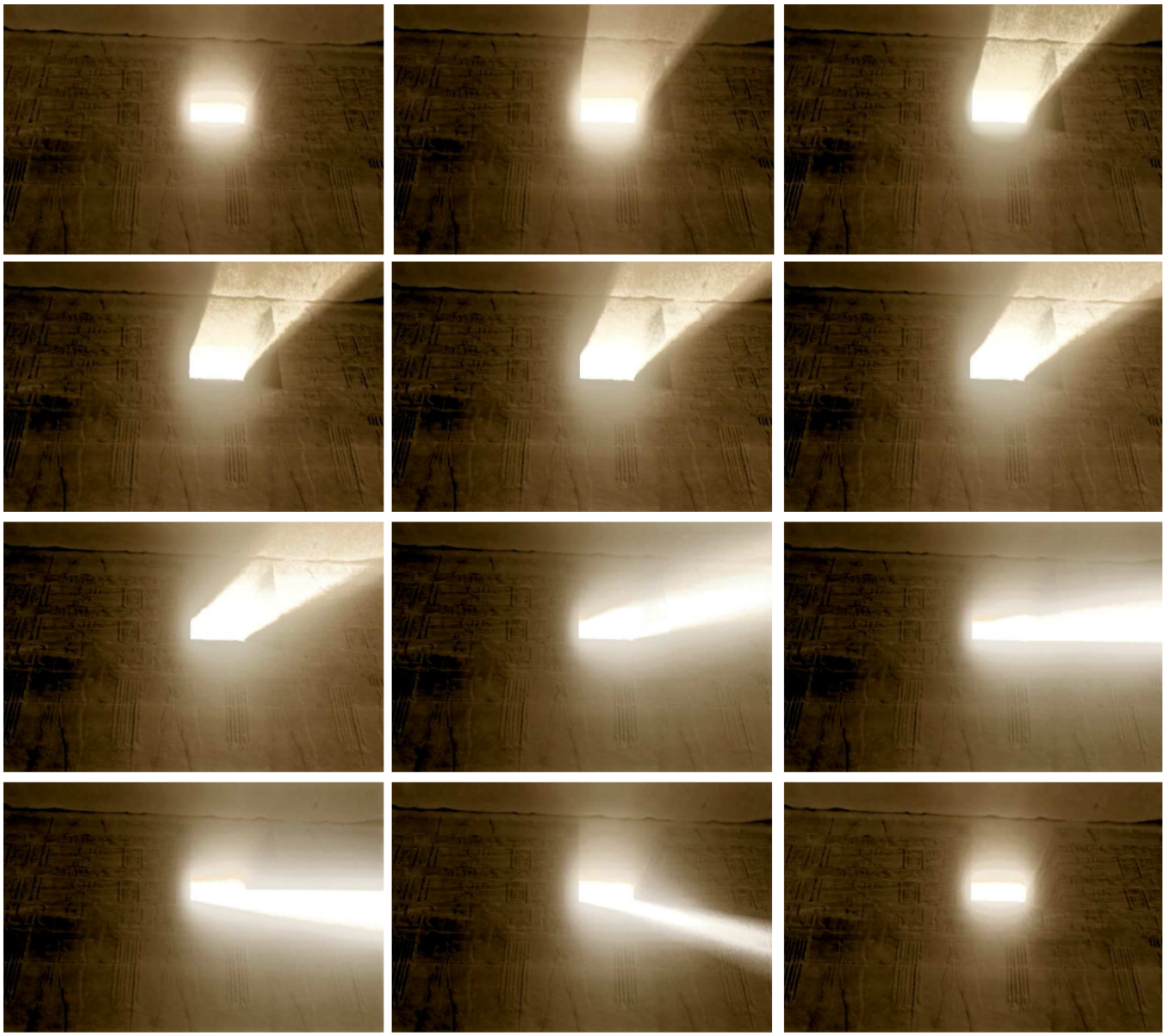


Figure 89: Sunlight entering through one of the narrow slits.

fail to do so under more general conditions. Second, of course, we need faster computations, for which a combination of a killer hardware architecture and smart software algorithms is needed.

18.1 The Quest for an Über-Model

Early work in perceptually assisted rendering was mainly concerned with the acceleration of ray tracing. Although techniques based on visual attention had been developed before, such as adaptive sampling [Mit87] for ray tracing which applied perceptual considerations, they have become more popular recently. An excellent overview of early perceptually-driven radiosity methods is given in [Pri01]. A frequency based ray tracer using a more complete model of the human visual system which incorporated the visual system's spatial processing behavior and sensitivity change as a function of luminance was developed by [BM95]. An even more sophisticated model incorporating visual masking was developed by [FPSG97].

Visual difference predictors have been used both to direct the next set of samples within stochastic ray tracing, and as a stopping criteria [Mys98, BM98]. These algorithms both required repeated applications of the perceptual error metric which was an expensive operation. The cost of such metrics were reduced in [RPG99] by precomputing the spatial frequency component from a cheaper estimate of the scene image. Spatiotemporal sensitivity of the human visual system was later added to Daly's Visual Difference Predictor to create a perceptually-

based Animation Quality Metric (AQM) [Mys02], which was used to guide a hybrid ray tracing and Image-Based Rendering (IBR) approach to improve rendering performance in a keyframe based animation sequence. Myskowski's framework assumed that the eye tracks all objects in a scene and the AQM added computational overhead to the rendering process.

Later a visual attention model was used in [YPG01] to improve the efficiency of indirect lighting computations for dynamic environments. Yee et al. exploited a saliency model termed the Aleph Map to adjust the search radius accuracy of the interpolation of irradiance cache values.

In [DPF03] a perceptual metric for interactive applications was presented in order to reduce the memory required for each diffuse texture while keeping the best image quality. Another perceptual metric was used in a component-based rendering framework to predict the relative importance of each component as a function of the materials visible from the desired viewpoint [SFWG04].

Saliency maps and the notion of task objects were used in a real time renderer to identify the most salient objects for which to render the glossy and specular components [HMYS01]. In [CCW03, SDL*05] both task maps and saliency map were used to vary a number of rays shot per pixel in a global illumination environment.

As we have seen, all these models, plus all the work done on tone mapping using characteristics of human perception, can be seen as partial models that work best under certain assumptions or specific tasks, but fail to do so under more general conditions. There is still lots of work ahead, most of it is not even directly related to computer graphics, but to the study of the physiology of the human body.

18.2 A Perceptual Model for Participating Media

Just to add one more colorful dot to the Impressionist mosaic of perceptual models (so much for our quest for an über-model!), we are currently working on a perceptually assisted algorithm for faster rendering of scenes with participating media. Whereas the results are far from final yet, preliminary tests suggest up to one-order-of-magnitude faster computations times, with minimum or even negligible perceptual degradation. The general idea is to precompute maps showing the areas of greater visual interest in the scene, taking into consideration the extinction and scattering of light in the medium.

Figure 90 shows two of our test scenes, named Lucy and Road. The leftmost image shows the gold standard for Lucy, rendered using ray marching, with 16 rays per pixel. To its right, the image rendered with the precomputed maps. The top Road image shows the gold standard for that scene, with the faster render shown below. The speedup factor for both scenes was slightly over 10. The maps for the Road scene are shown in figure 91. The method is currently being implemented in our in-house renderer *Lucifer* [GMAS05].

As we have seen, it is important to validate the resulting images using subjective responses from human observers. A psychophysical experiment was run with 48 human observers. For some scenes (Lucy) the results with perceptual strategies were indistinguishable from the gold standard, although for other scenes (Road) the participants could tell the gold standard from the one using precomputed maps. An important conclusion of this is that, even though the average pixel errors for all scenes were less than one percent, that does not directly mean that an image will be indistinguishable from a gold standard, as previous experiments had suggested. Whether or not the loss in quality in exchange for faster computation times is reasonable will depend on the particular application of the simulation. This nevertheless shows the importance of using human observers in the validation process of realistic image synthesis algorithms.

19 Acknowledgments

Kudos to Francisco Seron, and my two PhD students and fellow Assistant Professors, Oscar Anson and Adolfo Munoz, who have collaborated in most of the research presented in this section of the course, and to Veronica Sundstedt for her contributions on the simulations of the temple of Kalabsha, the state of the art in perceptually-guided rendering strategies and the perceptual strategies for rendering participating media.

Figure 90: Lucy, left: gold standard. Lucy, right: rendered using the XS-map. Road, top: gold standard. Road, bottom: rendered using the XS-map.

Figure 91: The different precomputed maps used for the Road scene.

Part VII

Collaboration in Virtual Environments – *Mashhuda Glencross and James Marsh*

Collaborative virtual environments promise both the possibility of social and work collaboration between geographically separated users. The idea of a shared virtual space were people can meet and discuss design problems, plan and play out training scenarios, play games has led to the development of a number of software frameworks to facilitate this process. Compelling collaborative environments must look, sound, behave and even feel as real as possible.

However this is challenging, there is still an unsolved barrier to providing consistent behaviorally rich shared virtual spaces. The impact of network latency (the delay between the transmission of a message and its reception) and jitter (the variation in latency) commonly found on the Internet causes significant problems for maintaining consistency. The inherent delay in networks means that not all people in the shared space may have a consistent view of the actions of others. We will show later in these notes that most techniques to manage latency and jitter involve a trade-off between consistency and responsiveness.

Of all the interaction modalities employed in multi-modal 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. Haptic updates must be performed within 1KHz update rate for the correct perception of solid contact [Bur96]. Therefore, we focus on potential pitfalls which arise in this case, and how to begin to mitigate them.

A number of experimental studies 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. A survey of these is presented by Glencross et al. [GOC05] Furthermore, the addition of haptic feedback as a means for input and constrained motion can lead to more user-centric interfaces [KW03].

20 Main Challenges for Haptic Collaboration

The benefits offered by haptic interaction in shared virtual environments are not 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 required 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 §21.1 we present the types of interaction modes possible in shared haptic virtual environments and the application contexts within which these can be employed.

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

20.1 Shared Manipulation

Two or more participants manipulating the same entity poses a serious 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 synchronization 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 must be employed in order to solve simultaneous constraints on motion, and this inevitably increases the complexity of the simulation. Many simple collaborative haptic demonstrators employ fairly basic linear force models [BHSS00, SRGS00, Hub02, OMJ*03, MSP*03, SBNG03, HHSW04, KKT*04, GHSA05] 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 between users can 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 major challenges detailed later.

20.2 Distribution over the Internet

Since there is no way to entirely remove network latency, it is difficult to ensure that all participants in a shared virtual space always see exactly the same thing at the same instant in time. This leads to a control issue, in particular if state in the environment is shared across all participants then where is the most appropriate place for control to reside. As we will show later in these course notes, at the heart of all distribution architectures is a choice relating to who has control.

Due to haptic update and consistency requirements few state-of-the-art collaborative haptic systems operate over a real WAN such as the Internet. There are few studies in which this has been attempted [OMJ*03, HHSW04, KKT*04, GHSA05], 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 synchronize simulations as well as mitigate the effects of network latency.

Techniques for synchronizing state across multiple participants on a WAN are an important consideration for collaborative virtual environments and depend primarily on distribution architecture choices. For behavior-rich multi-sensory environments these issues become even

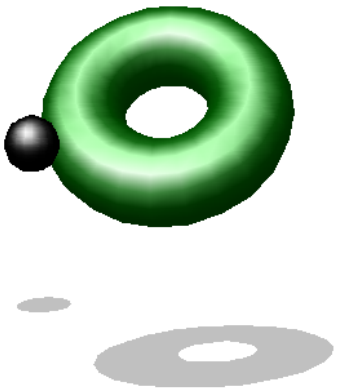


Figure 92: User colliding with an entity

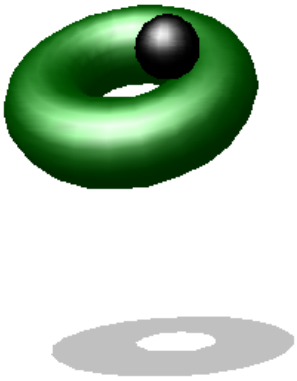


Figure 93: User touching an entity

more crucial as potentially complex state information has 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.

21 Shared Interaction

21.1 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 categorization further, in order to detail the types of complex interactions possible.

21.1.1 User-Entity

User-entity interactions are those which occur between a user and any specific entity in the environment. Figure 92 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 93.

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 94 (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.

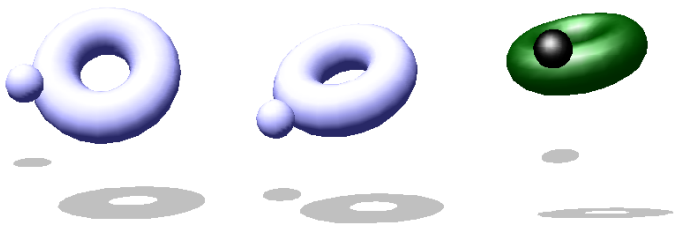


Figure 94: User moving an entity

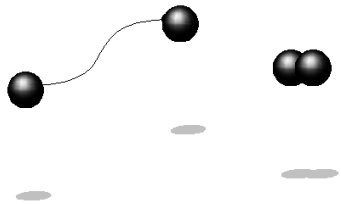


Figure 95: User touching or colliding with another user

21.1.2 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 95. 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 96. Applications have adopted anything from text based communication to 'Voice Over IP'.

21.1.3 User-Entity-User

Interactions in the category user-entity-user can potentially be sophisticated. Firstly, a user colliding with an entity can cause it to move and in turn collide with another user as shown in Figure 97. 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 98. 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 99 where two or more participants manipulate the same entity is possibly the most complex interaction situation possible. To achieve this requires that state is synchronized between participants in addition to collision detection and

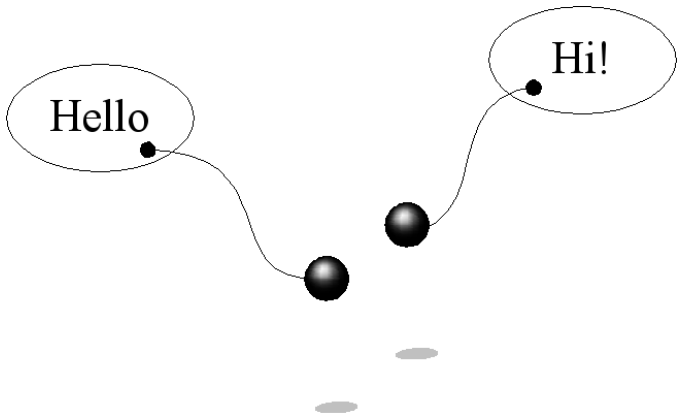


Figure 96: User talking with another user

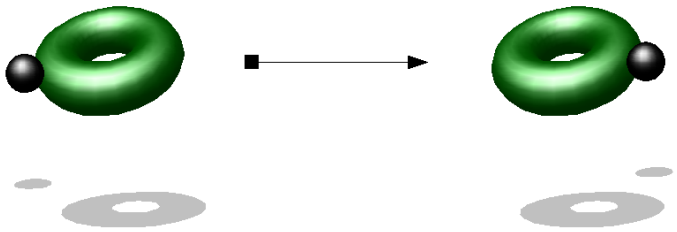


Figure 97: User colliding with entity which in turn collides with another user

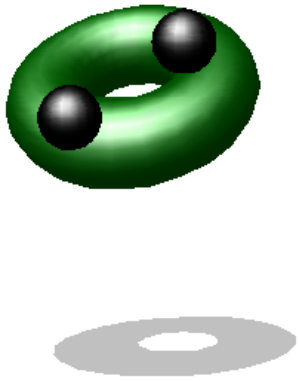


Figure 98: Two users touching the same entity

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.

21.1.4 Entity-Entity

Entity-entity interactions of the type shown in Figure 100 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 limitations (algorithms for this are detailed in §28).

An entity exerting a force on another entity, as shown by the curved line between the torus and cone in Figure 101 may exhibit complex behavior 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 synchronized between all participants if a meaningful collaboration is to be undertaken.

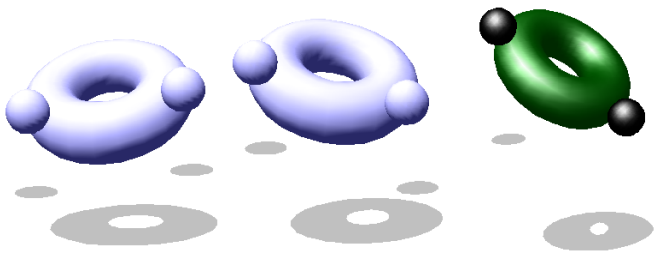


Figure 99: Two users moving the same entity

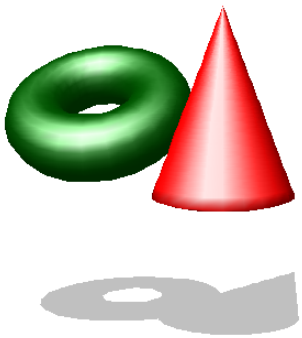


Figure 100: An entity colliding with another entity

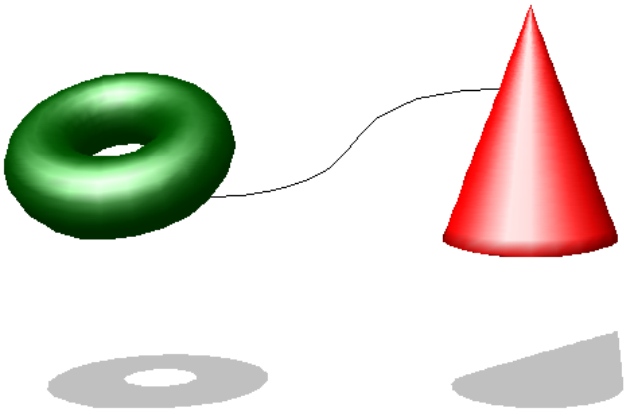


Figure 101: An entity exerting a force on another entity

21.2 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 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.

21.2.1 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.

21.2.2 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 maneuver 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 *Alan* is demonstrating an assembly maintenance operation to a group of users but *Mashhuda* has some difficulty understanding the sequence; she may choose to take hold of the entity *Alan* is manipulating, in order to intervene, and pause the sequence.

21.2.3 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].

22 Distributing Simulations

State information relating to a shared simulation (for example positional updates, forces, constraints, changes in size etc.), together with user induced changes have to be transferred over a network to all connected participants. The delay in transmission over a network will have very different consequences on the consistency of a simulation depending on the network distribution strategy employed.

When sharing simulations between participants over a network, each user should ideally experience at least a plausible view of the simulation. This becomes even more critical in applications in which users may communicate about and interact with simulations changing their properties. If each person does not experience a consistent, synchronized and plausibly correct simulation of behavior, it is difficult for people to undertake a meaningful collaborative task in which they all influence the same rigid or deformable entity. The types of problems that occur due to network latency and jitter are detailed in the following sections.

22.1 Injection Problems

Figure 102 illustrates what happens when two people, each running their own instance of a simulation exchange events [MGPH06]. These events from each user are inserted *injected* into the remote person's simulation at intervals corresponding to different network event delivery characteristics.

The initial condition shown in Figure 102(a) occurs over a low reliable fixed latency network connection. This type of event delivery characteristic may be typically found on a local area network. Events from each participant are delivered quickly and in time to be taken account of in the remote simulations and consequently both remain consistent with each other.

In the second condition 102(b), the simulation occurs over a reliable but high fixed latency connection. This type of network event delivery characteristic may be typical of good quality academic networks. Each participant's events are delivered consistently late, and so the order in which these are incorporated into the simulations differs but in a predictable manner. This predictability means that approaches to buy some extra time to await the remote events may be employed to ensure the same event ordering between simulations.

The third condition is much more complex and represents a variable latency network connection 102(c) This type of event delivery characteristic is commonly found on the Internet. Notice that in this case the order in which events are introduced into the simulations is much more haphazard and difficult to predict.

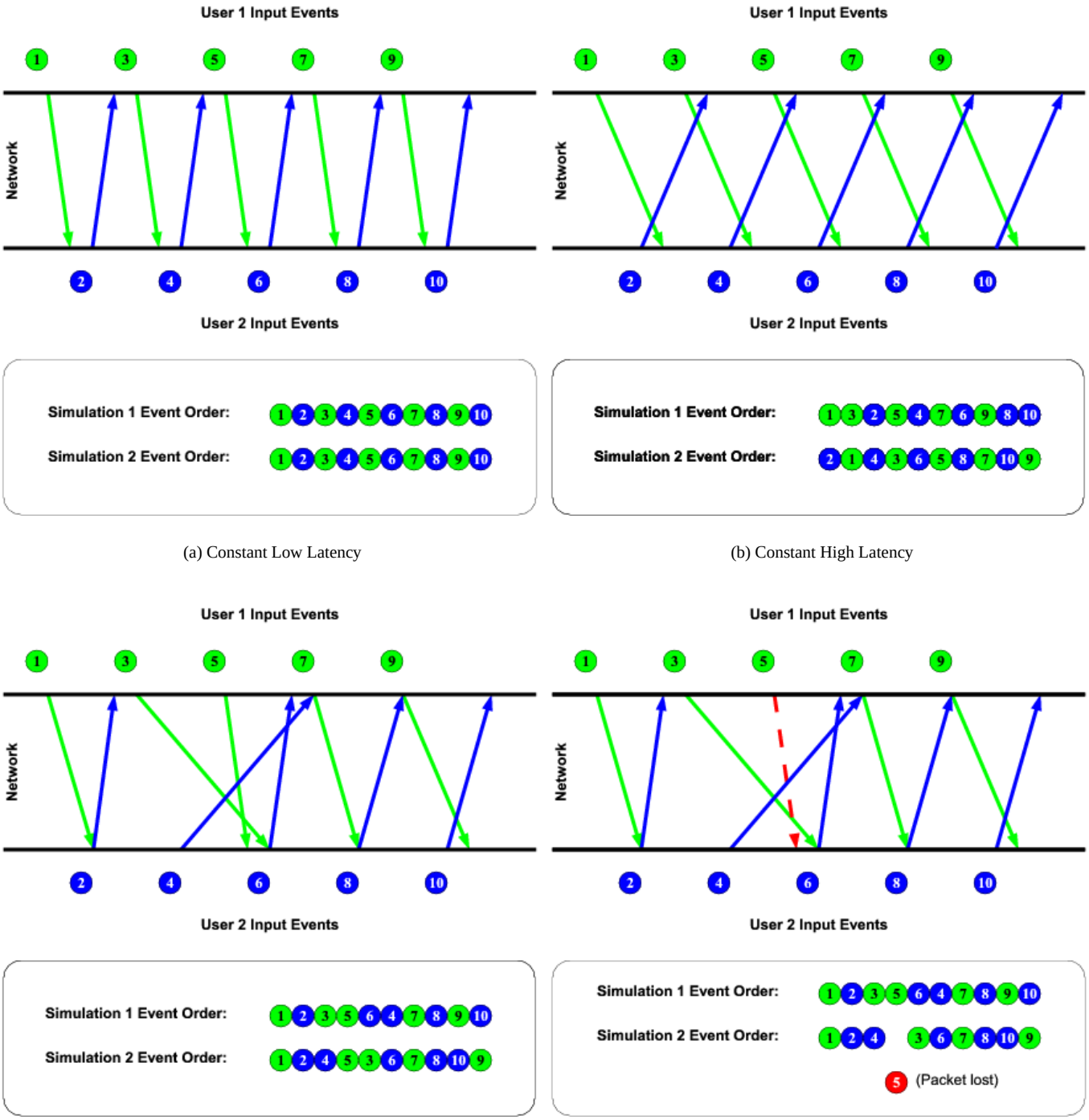
Finally the most catastrophic case 102(d), in which the latency is both variable and events may even be lost, results in substantially different simulation outcomes for both participants. This worst case event delivery characteristic is also commonly found on the Internet, normally between small to medium sized organizations. Depending on the network protocol used, this loss of events may be impossible to detect. TCP/IP guarantees to deliver data by transparently re-sending lost updates but incurs 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. Depending on the type of simulation shared between users, this can have serious consequences on the outcome of each simulation [MGPH06].

22.2 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 recognizes 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 behavior 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 on the other hand a collaborative CAD application employing rigid body dynamics simulations. If two or more participants are required to co-operatively lift, rotate, and position a 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 to this question depends both on the network conditions



(a) Constant Low Latency

(b) Constant High Latency

(c) Latency and Jitter

(d) Latency, Jitter and Packet Loss

Figure 102: The effect of different network conditions on distributed simulations [MGPH06]

Destination	Distance (km)	Lost Packets	Round-Trip Time (ms) Min.	Max.	Mean	Mean Deviation	Mean Round-Trip Update Rate (Hz)	Typical 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 [MGPH06]

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.

22.3 Synchronizing State

The problem of divergent simulations is solved by synchronizing state information across all participants. This is normally 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 synchronized 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.

23 Real Networks

Real wide area network characteristics vary widely but can be typically characterized in terms of two metrics, the amount of latency and jitter. Latency (the delay between a message being transmitted and received) 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. Most people are familiar with latency encountered in long distance telephone conversations, when faced with such communications delay, in structured conversations people may adopt a *wait and talk* (turn-taking) strategy or more often curtail the conversation [EM63, RK63]. Latency can also 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.

23.1 Network Latency and Jitter

The aim of any distribution policies facilitating collaboration, is to present a consistent and coherent view of a shared environment to each person [MGPH06]. Ultimately the speed of light imposes a finite upper bound on how fast information can be exchanged across a network, and in practice real network performance is well below this level. Consequently, regardless of architecture, two remote users cannot share an instantaneous, fully-synchronized 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 103 to 108 present 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 distribution choices appropriate for a local area network will not necessarily scale for typical wide area networks such as those profiled, and likewise applications designed specifically to work over wide area networks are unlikely to take full advantage of the faster local environment [MGPH06].

23.2 Impact on Task Performance

A large 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.

The effects of latency in the visual modality [Mic63, MW93, PW02, MRWB03, MGP*04] have been studied in much greater detail than the haptic modality. Preliminary research in the latter has provided mixed results with some studies suggesting that users find it difficult to

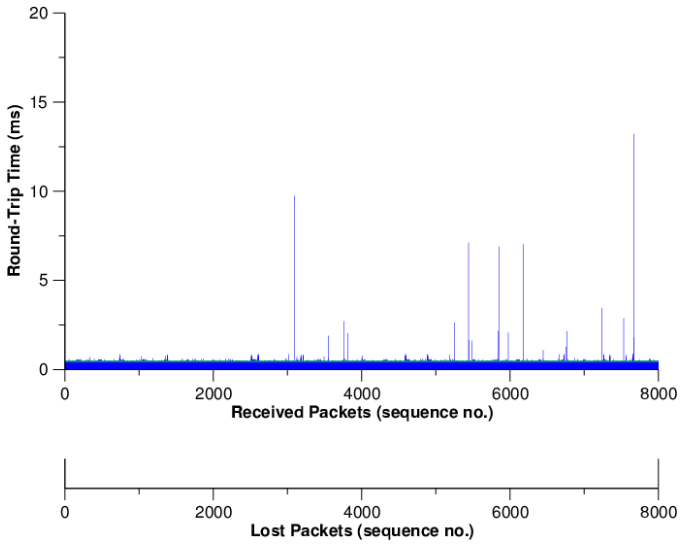


Figure 103: Measured performance on The University of Manchester local LAN [MGPH06]

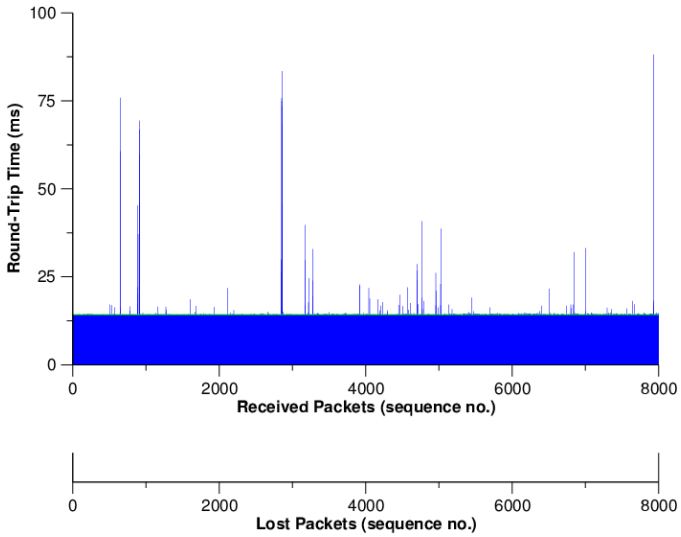


Figure 104: Measured performance to The University of Bristol, United Kingdom [MGPH06]

perceive haptic latency and others suggesting that latency in haptic feedback quickly impacts upon task performance [Fer66, AH03, WTR*03, AZWS04, JH04, JH05, MGPH06]. This discrepancy is likely to be due to a number of factors related to the task studied, the devices used in the studies, and the distribution policy employed.

In the visual modality, users 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 behavior / 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. 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.

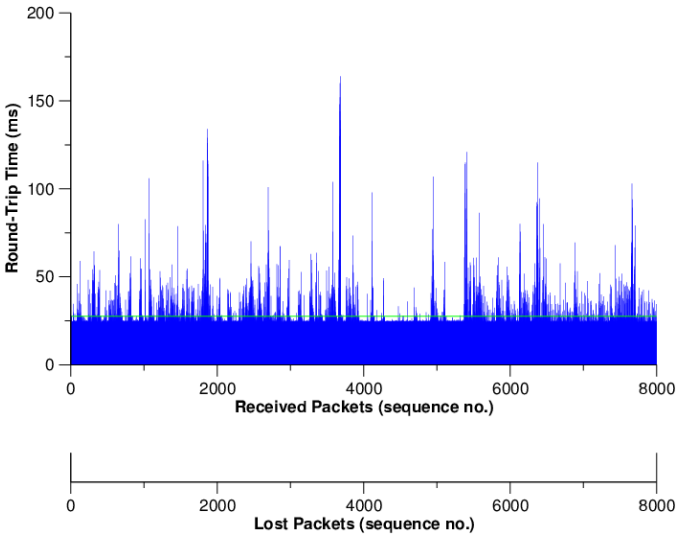


Figure 105: Measured performance to FCS, Amsterdam, The Netherlands [MGPH06]

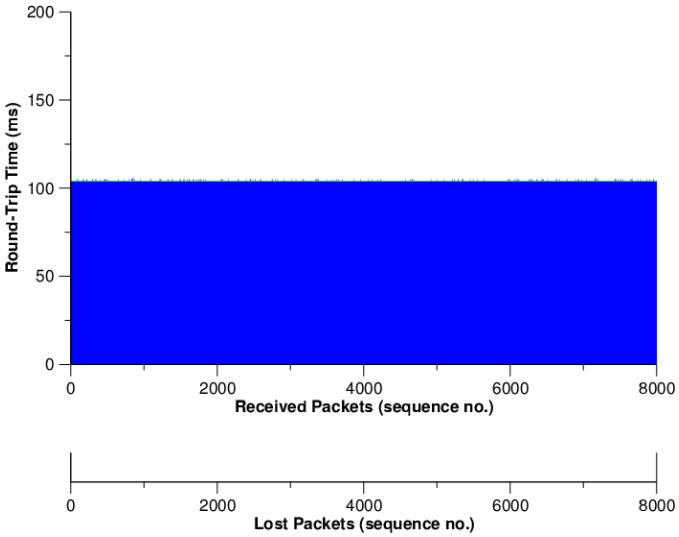


Figure 106: Measured performance to The University of North Carolina, USA [MGPH06]

23.3 Architectures for Distributed Systems

The two main architectures adopted for distributed systems are client-server and peer-to-peer, each of these have their relative 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.

23.3.1 Client-Server

The client-server architecture (shown in Figure 109) is one of the most simple architectures to use for supporting consistent simulations. It uses a centralized server that runs a simulation of the virtual environment and communicates relevant state changes to all connected clients. Managing state across all clients is easy because the server alone maintains state and clients have to wait for the latest information to become available. Clients normally contain a representation of the 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 communicating events and injecting them into separate simulations, information describing the outcome of a single centralized

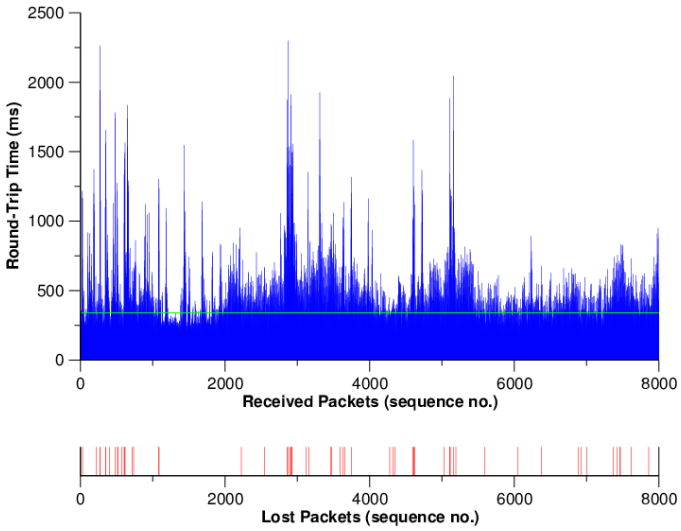


Figure 107: Measured performance to Labein, Bilbao, Spain (afternoon) [MGPH06]

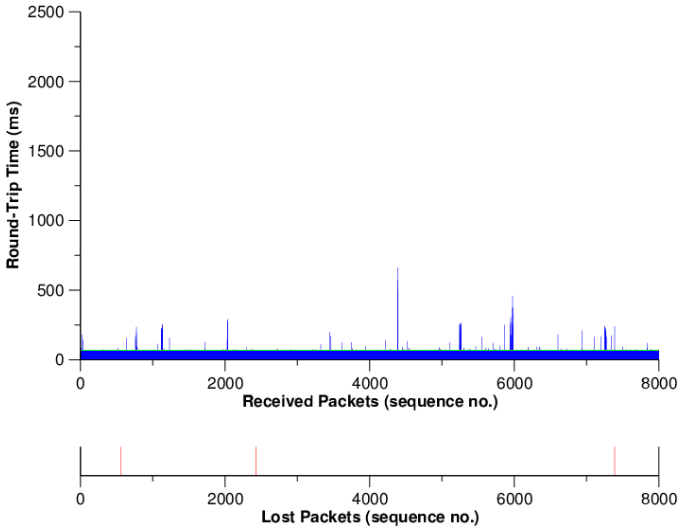


Figure 108: Measured performance to Labein, Bilbao, Spain (night) [MGPH06]

simulation is transmitted by the server 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 program. The client may briefly display an inconsistent view of the world, but this is quickly corrected when a new update does eventually arrive. In addition to the synchronization 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, MGPH06].

The biggest disadvantage of this 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. This means that from the user's perspective the environment seems slow to respond to changes. The architecture trades local responsiveness for guaranteed consistency. It is generally accepted that this approach 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 a local area network, 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 [MGPH06].

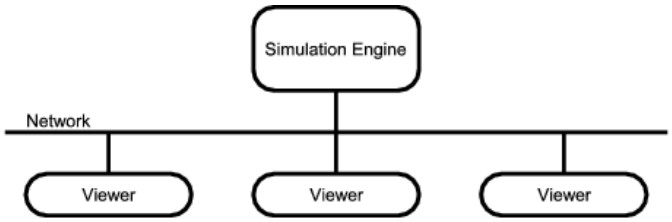


Figure 109: Client-server architecture [MGP*04]

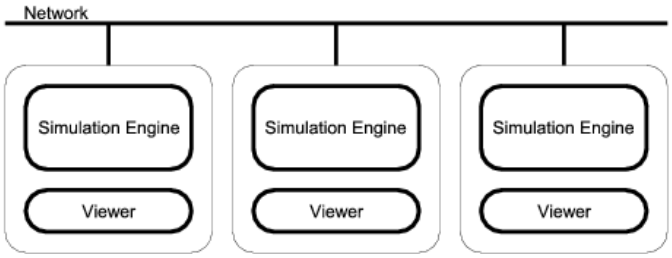


Figure 110: Peer-to-peer architecture [MGP*04]

23.3.2 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 110. 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 synchronization 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 [MGPH06].

However, in complex environments synchronization 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 are few obvious ways to correct the causal inconsistencies that arise [MGPH06].

Peer-to-peer architectures are most successfully employed on networks with an update delivery rate similar to that shown in Figure 102(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 cause a significant synchronization problem [MGPH06].

23.3.3 Shades In-Between

Due to the limitations of the two main distribution architectures detailed above, a variety of hybrid approaches have been employed. A third configuration commonly used attempts to combine the low-latency advantages of a peer-to-peer network, with the synchronization benefits of a client-server architecture. This is achieved through a peer-to-peer system which enforces object-based locking as illustrated in 111 (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 simulated entity at any instant in time, however it also prohibits simultaneous co-operation.

For many applications this architecture will be sufficient, but it imposes strict turn taking interaction which may be unintuitive in certain application contexts [MGPH06]. 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 102(a). An architecture similar to this was suggested by Hespanha et al. [HMS*00].

An alternative architecture proposed by Marsh et. al. [MGP*04, MGPH06] utilizes a number of distributed simulation servers, typically with one per local network as shown in Figure 112. 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.

Upon starts up, all of the simulation engines are dormant. Clients connecting to the environment profile the network link to each available simulation server, in order to determine a preferred choice 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.

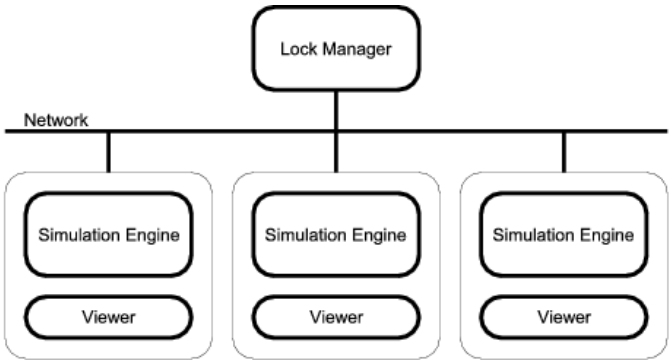


Figure 111: Enforced turn-taking (locking) architecture [MGP*04]

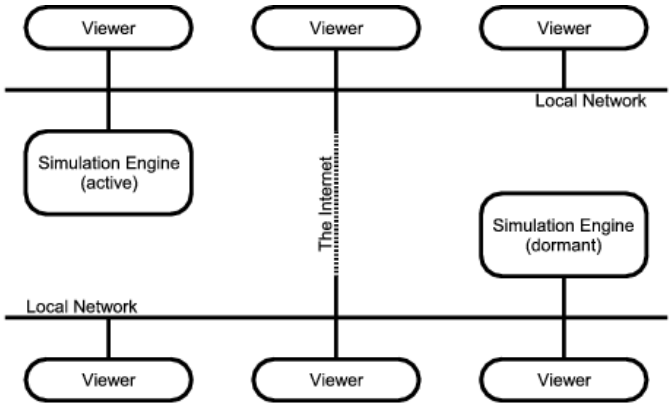


Figure 112: Roaming server architecture [MGP*04]

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 manipulate the same simulation entities. However the second user may suffer greater latency than the first depending on their geographic location and network connection.

24 Considerations for 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.

24.1 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 synchronization issues rapidly begin to cause problems in peer-to-peer systems with heavy simulation synchronization needs. Thus for behavior-rich environments, this architecture is only really suitable to employ on fast networks.

Computation of costly multi-body simulations is often parallelized. However synchronization issues present a significant challenge, even to shared memory supercomputers, where communications latency is many orders of magnitude less than that on a local area network. 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 introduce further latency [MGPH06].

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 resynchronize updates such that they are incorporated as in Figure 102(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 [MGPH06].

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 behaviors 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 [MGPH06].

24.2 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 synchronization demands. The client-server approach is easier to implement from the point of view of provision of synchronization 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 minimizing 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.

24.3 Exploiting Perception

Recall that Deva's [PCMW00] object/subject model for distributing rich behavior decouples the real state of entities from the perceptual representation. The system's authors recognized 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 behavior, that define their semantic roles. User's in the environment experience this behavior, by observing the changes in an entity's directly perceivable properties. Therefore, if an entity stores attribute internally as part of its simulated behavior, 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 visualizers. 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 practice 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 behavior 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 synchronization 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].

24.3.1 Graphical, Behavioral and Haptic Correspondence

Systems incorporating high quality graphics, complex behavioral simulations, and haptic feedback often need to maintain different data relating to each. In shared virtual environments this graphical, behavioral 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 behavioral 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].

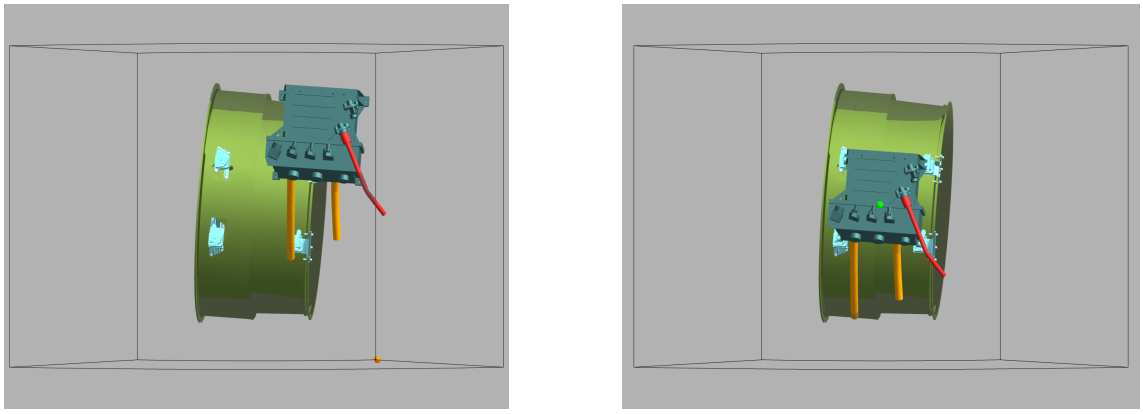
The latter technique exploits the fact that in many cases, graphical feedback plays a dominant role in user perception of the environment. Consequently as long as haptic feedback supports and re-enforces user expectations from the graphical display, the correspondence between underlying models need not 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 §35.

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].

25 Collaborative Virtual Prototyping

We show a new system for facilitating haptic collaboration that was tested between the University of North Carolina at Chapel-Hill and the University of Manchester. The system uses a perceptually inspired approach to maintain consistency and responsiveness, within acceptable tolerances, between participants undertaking a shared manipulation task.

Figure 113 shows the test task undertaken. Two users were required to collaborate to position the fuel box from the start position 113(a) to the target position 113(b).

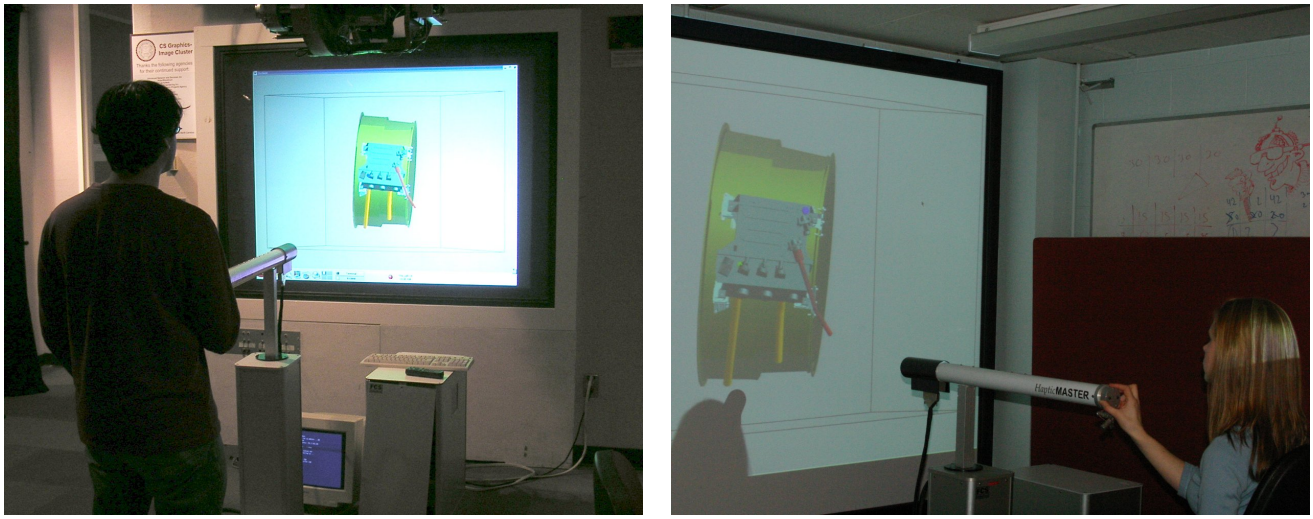


(a) Start position of the box

(b) Target position of the box

Figure 113: Start and target position in the cooperative CAD prototyping study.

The images shown in Figure 114 were taken during actual testing between the USA and UK sites. Each site was equipped with the same type of haptic device and a large projected screen.



(a) User in the USA

(b) User in the UK

Figure 114: Two geographically distant users performing a cooperative haptic prototyping task. The user on the left is in the USA and the user on the right in the UK. These photographs were taken during an actual transatlantic trial.

Figure 115 shows two users each with an identical equipment set-up performing the task at the UK site but with simulated latency equivalent to that between North Carolina and Manchester.

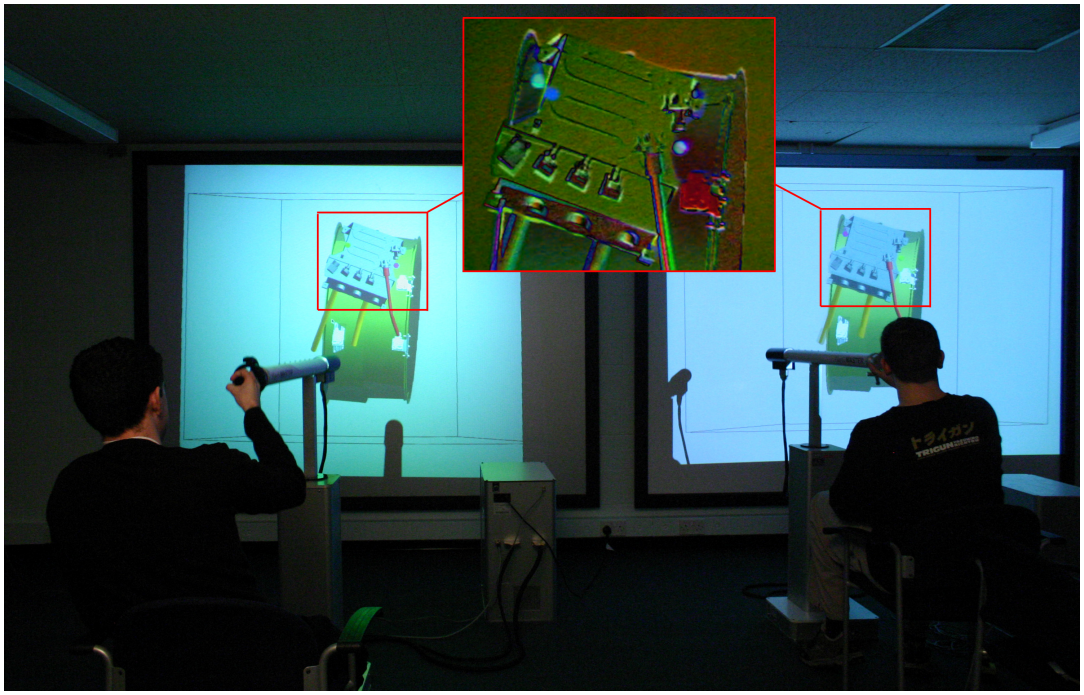


Figure 115: Two users performing the task with simulated latency equivalent to the WAN condition and synchronization/prediction.

Acknowledgments

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 Greg Ward for his comments and our research 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; Matt Aranha at The University of Bristol for invaluable help with profiling typical networks. Finally we would also like to thank Mary Whitton, Jeff Feasel, Luv Kohli and Fred Brooks at the University of North Carolina for invaluable help with collaborative haptic tasks over the Internet.

Part VIII

Real-time Interaction in Virtual Environments

26 Interaction in 3D Environments

We briefly describe typical devices used in interactive virtual environments to provide more intuitive input and display.

27 Types of Input/Output Devices

The range and configuration of available input and output devices that can be used in virtual environments is substantial. We refer the reader to Virtual Reality Systems [Vin95] for further information. Here we classify the devices we briefly discuss in terms of Tracking, Graphical Display, Haptic Display and Auditory Display.

27.1 Tracking

Tracking technologies aim to find the position and orientation in space of either the user's head, hand or even full body so that it can be used and re-produced in a virtual environment.

27.1.1 Head Tracking

Head mounted displays (shown in Figure 116) are commonly used to isolate and immerse users in virtual environments. A simple action such as turning one's head changes what we see in the real world. Similarly, to present correct graphical display to the user, 6 degree of freedom position and orientation information providing the pose of the user's head must be made available.

Figure 116: Example Head Mounted Display

A plethora of head tracking technologies exist randing from those based on gyroscopic, magnetic, and optical tracking technologies. The HiBall Tracking System [WBV*01] developed at the University of North Carolina is considered to be one of the most accurate.

27.1.2 Full-hand(s) Gesturing

Full hand tracking offers the potential for virtual representations of a user's hand to be correctly posed in a virtual environment. Furthermore, it provides the possibility of more intuitive human computer interaction. Gestures may be recognized from hand pose and used to allow natural user interaction in navigation tasks for example. One possible way to obtain the pose of the user's hand is to use a sensor glove (shown in Figure 117) that is worn by the user and reports positional information of the hand and fingers.

Figure 117: Example of a Sensor Glove

27.1.3 Full-body Tracking

Real-time tracking of the full body is the most complicated, and is the subject of much ongoing research. In particular non-intrusive methods based on sequences of images from cameras offer a good way to re-produce the user's entire body pose in a virtual environment. However to achieve highly accurate information in real-time, currently remains an unsolved problem.

27.2 Graphical Display

Figure 118 shows a user interacting with virtual objects shown graphically (at the top) and their actions in the real world (shown at the bottom). Notice there is a graphically simple representation of the user's hand in the virtual world and some simple visual feedback is given while the user preforms their task.

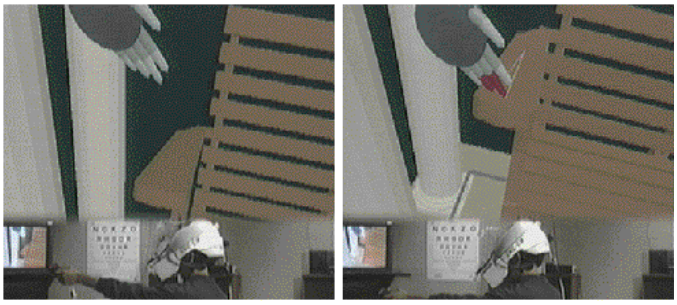


Figure 118: User interacting with virtual objects

27.3 Auditory Display

Headsets providing spatialized auditory display (as shown in Figure 119(a)) are often used in virtual reality application. In the real world, we hear sounds created by a vast number of sources. Simulating such complex auditory feedback in real-time is challenging. Figure 119(b) shows an image of hundreds of rings falling onto a wooden table, each creating a suitable sound in real-time as it collides with the table. We detail the physically based model for this in §29.

(a) Vision Technology

(b) Physically based sound simulation

Figure 119: Auditory display

27.4 Haptic Display

Haptic displays currently available range from simple vibrotactile pads (such as those used in cell-phones) to three (position) and six degree of freedom (position and orientation) force feedback devices. Recently, a number of low-cost three degree of freedom haptic displays have become available. Some examples of haptic devices are shown in Figures 120.

(a) Sensable Technologies

(b) Immersion

(c) Northwestern University

Figure 120: Haptic display

Collision detection poses a particular problem for haptic interaction with polygonal models of objects in virtual environments and techniques must determine interaction between objects at interactive rates. For haptic and auditory display, it is necessary to simulate physical interaction between objects and this involves carrying out collision and proximity queries. Algorithms for these depend on the number of degrees of freedom of the interaction and are outlined in the following sections.

28 Introduction to Collision Detection Techniques

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

In this section, we first describe the problems of interference detection and computation of separation distance between polyhedra, with an emphasis on algorithms specialized for convex polyhedra. Then we survey the use of hierarchical techniques, algorithms for the computation of penetration depth, and multi-resolution collision detection.

We begin by briefly presenting the Collide system Architecture shown in Figure 121.

During the sweep and prune stage shown in Figure 122, the following steps are undertaken:

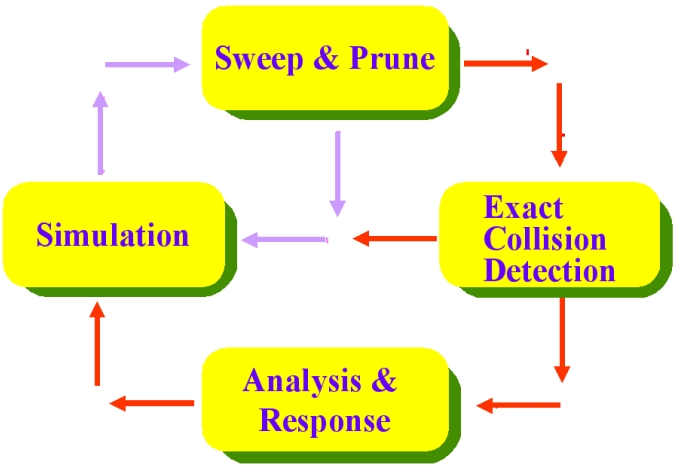


Figure 121: Collide system architecture

- Compute the axis-aligned bounding box (fixed vs. dynamic) for each object
- Dimension Reduction by projecting boxes onto each x, y, z axis
- Sort the endpoints and find overlapping intervals
- Possible collision – only if projected intervals overlap in all 3 dimensions

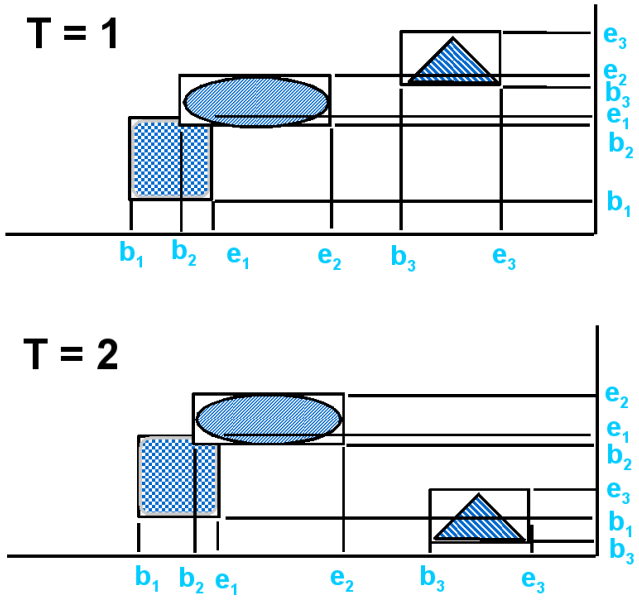


Figure 122: Sweep and prune

For dynamic simulation where the velocity and acceleration of all objects are known at each step, we use the scheduling scheme (implemented as heap) to prioritize "critical events" to be processed. Each object pair is tagged with the estimated time to next collision. Then, each pair of objects is processed accordingly. The heap is updated when a collision occurs.

To estimate the time to collision is shown in Equation refetc-equation:

$$a_{max} = |a_{lin} + \alpha \times r + \omega \times \omega \times r| \tag{25}$$

$$v_i = |v_{lin} + \omega \times r| \tag{26}$$

$$t_c = \{\left(v_i^2 = 2a_{max}d\right)^{1/2} - v_i/a_{max}\} \tag{27}$$

Where:

- a_{max}: is an upper bound on relative acceleration between any two points on any pair of objects.
- a_{lin}: relative linear acceleration
- α : relative rotational accelerations
- v_{lin} : relative linear velocity
- ω : relative rotational velocities
- r: vector difference between the center of mass of two bodies
- d: initial separation for two given objects

28.1 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.

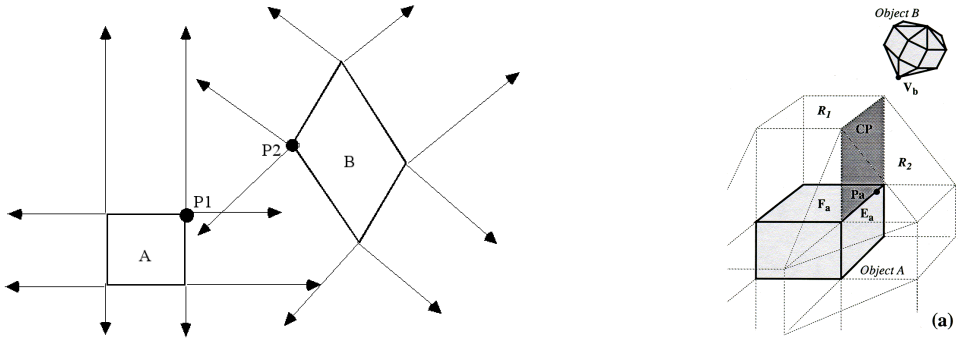
The separation distance between two polyhedra A and B is equal to the distance from the origin to the Minkowski sum of A and $-B$ [CC86]. This property was exploited by Gilbert et al. [GJK88] in order to design a convex optimization algorithm (known as GJK) for computing the separation distance between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

Lin and Canny [LC91, Lin93] designed an algorithm for computing separation distance by tracking the closest features between convex polyhedra. The algorithm "walks" on the surfaces of the polyhedra until it finds two features that lie on each other's Voronoi region. The following list summarizes the approach:

- Partition of space into cells such that all points in a cell are closest to only 1 primitive
- Geometric primitives are Voronoi sites
- Set of points closest to a particular site is a Voronoi region
- Set of all Voronoi regions is the generalized Voronoi diagram

The basic idea of the algorithm is that it exploits motion coherence and geometric locality, *Voronoi marching* runs in nearly constant time per frame, independent of the geometric complexity. Mirtich [Mir98] later improved the robustness of this algorithm.

Figure 123(a) shows a simple 2D example in which objects A and B and their Voronoi regions: P1 and P2 are the pair of closest points between A and B. Note P1 and P2 lie within the Voronoi regions of each other. Figure 123(b) illustrates feature tracking in 3D using Voronoi regions.



(a) Voronoi marching – 2D example

(b) Closest feature tracking

Figure 123: Voronoi marching

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].

28.2 Bounding Volume Hierarchies

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].

28.2.1 Methods of Hierarchy Construction

Assuming that an object is described by a set of triangles T, a BVH is a tree of BVs, where each BV C_i bounds a cluster of triangles $T_i \in T$. The clusters bounded by the children of C_i constitute a partition of T_i. The effectiveness of a BVH is conditioned by ensuring that the branching factor of the tree is $O(1)$ and that the size of the leaf clusters is also $O(1)$. Often, the leaf BVs bound only one triangle. A BVH may be created in a top-down manner, by successive partitioning of clusters, or in a bottom-up manner, by using merging operations. We briefly outline how to build an OBB tree in Figure 124.

In order to perform interference detection using BVHs, two objects are queried by recursively traversing their BVHs in tandem, as shown in Figure 125. Each recursive step tests whether a pair of BVs a and b, one from each hierarchy, overlap. If a and b do not overlap, the recursion branch is terminated. Otherwise, if they overlap, the algorithm is applied recursively to their children. If a and b are both leaf nodes, the triangles within them are tested directly. This process can be generalized to other types of proximity queries as well.

RAPID uses a tree of oriented bounding boxes (OBBs), PQP uses a tree of swept sphere volumes (SSVs) and SWIFT++ uses a tree of convex hulls.

28.2.2 Choices of Bounding Volumes

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.

28.2.3 Bounding Box Overlap Tests

We now briefly outline how to perform bounding box overlap tests. Figure 126(a) illustrates the separating axis theorem. L is a separating axis for OBBs A and B, since A and B become disjoint intervals under projection onto L. In Figure 126(b), L is a separating axis if $s > h_a + h_b$.

The basic steps to carry out bounding box overlap tests are:

- Project boxes onto axis. If intervals do not overlap, it is a separating axis
- A separating axis exists if and only if boxes are disjoint
- 2D OBBs: 4 axes to test
- 3D OBBs: 15 axes to test

The strengths of this overlap test are:

- 80 to 234 arithmetic operations per box overlap test
- No special cases for parallel/coincident faces, edges, or vertices
- No special cases for degenerate boxes
- No conditioning problems
- Good candidate for micro-coding

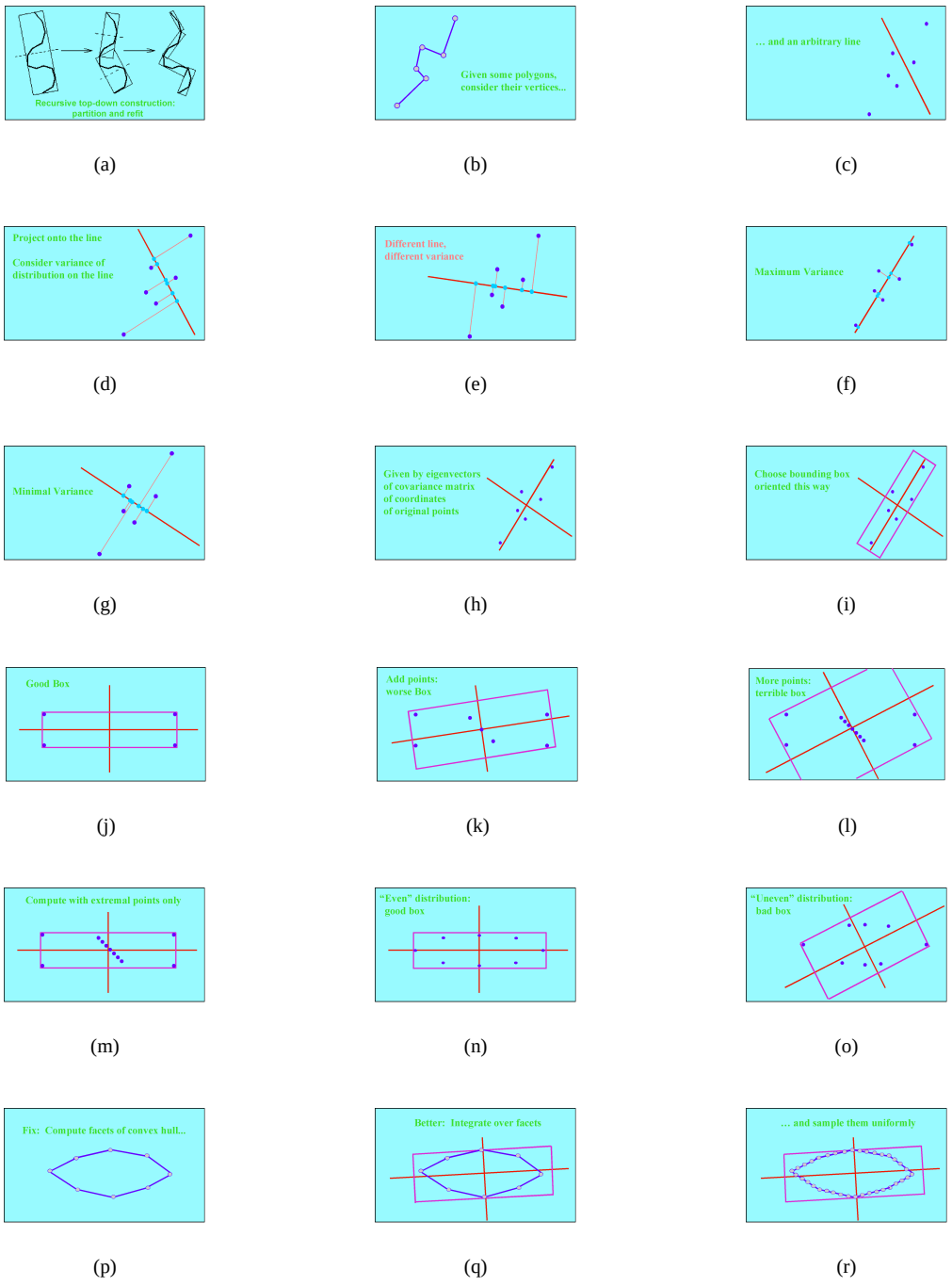


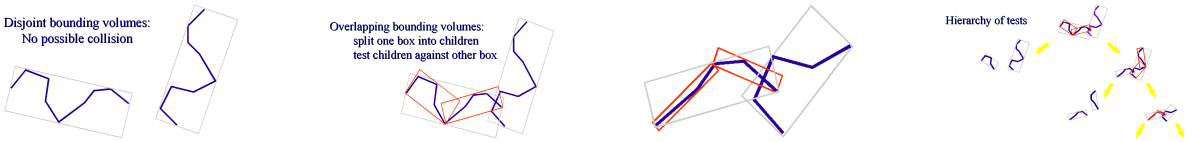
Figure 124: Building an OBB Tree

Figure 127 shows the relative performance of OBB overlap tests compared to Spheres and axis aligned bounding boxes.

28.2.4 Optimization Techniques

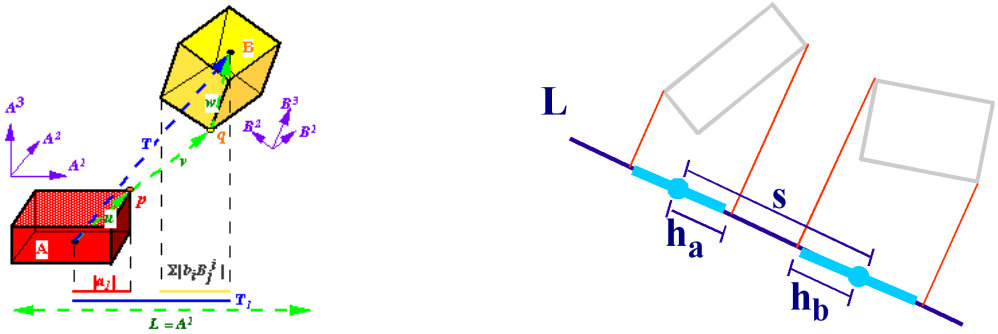
The general idea behind convex-hull based queries using surface-decomposition is:

- Convex surface decomposition. Partition surface into patches, then compute the convex hull for each patch to give pieces
- Bounding volume hierarchy (BVH) construction. Use pieces and convex hulls as bounding volumes



(a) (b) (c) (d)

Figure 125: Tree traversal



(a) (b)

Figure 126: Separating axis theorem

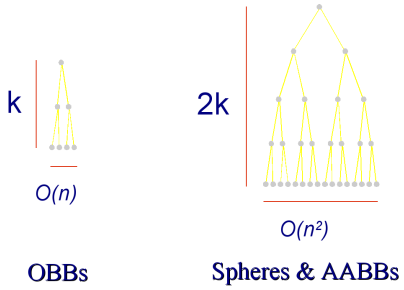


Figure 127: Performance: Overlap tests

- Hierarchical query. Use SWIFT to query between a pair of bounding volumes
- Coherence-based query acceleration. Piece caching, priority directed search and generalized front tracking

To perform the decomposition, it is necessary to traverse the surface using a breadth first search or a depth first search on the dual graph preserving the following criteria:

- Local convexity
- Global convexity
- Containment

Figure 128 shows top-down construction of a bounding volume hierarchy with various splitting metrics.

To perform a hierarchical query A pair of convex polyhedra can be tested against one another using the Lin-Canny closest features algorithm which can give the following output:

- Intersecting (yes/no)
- If intersecting return the features that intersected
- Else return the minimum distance and the nearest features

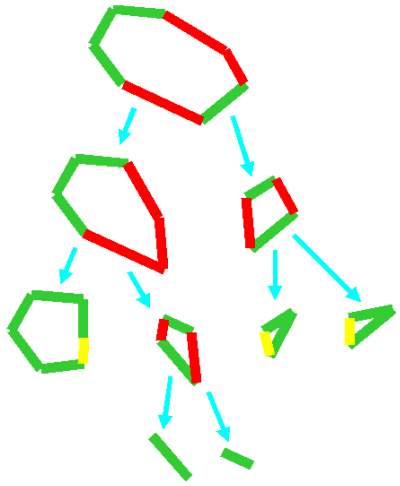
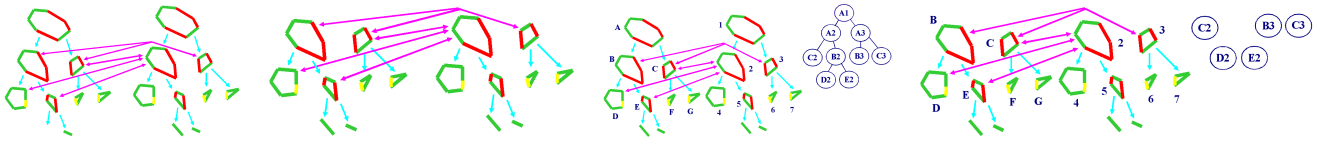


Figure 128: Bounding volume hierarchy

In order to perform piece caching it is necessary to remember nodes in the BVH that answered the last query, and start the current query using those nodes. For a priority directed search, we perform a distance computation acceleration and expand the pairs of nodes that have the smallest distances first. Generalized front tracking avoids traversal of the upper levels of the hierarchy and is shown in Figure 129. The next query has a head start, pair trees are used to keep track of this, the leaves of the pair tree are used to start the next query.



(a) (b) (c) (d)

Figure 129: Generalized front tracking

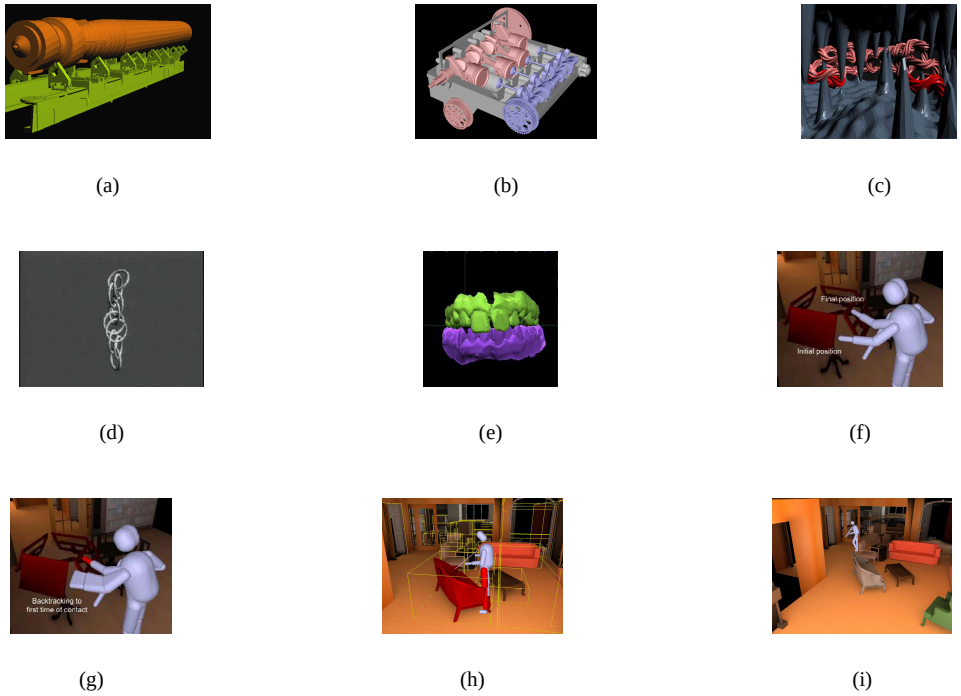
28.2.5 Application Demonstration

Example applications developed are shown in Figure 130.

29 Real-time Sound Simulation – *Nikunj Raghuvanshi and Ming C. Lin*

Most interactive applications today employ recorded sound clips for providing sounds corresponding to object interactions in a scene. Although this approach has the advantage that the sounds are realistic and the sound-generation process is quite fast, there are many physical effects which cannot be captured by such a technique. For instance, in a typical collision between two objects, the loudness and timbre of the sound is determined by the magnitude and location of the impact forces – a plate sounds very differently when struck on the edge compared to when it is struck in the middle. Consequently, if the collision scenario changes slightly, the sound exhibits a corresponding change. Such subtle effects can add substantial realism to a typical scene by avoiding the repetitiveness common to recorded sound clips. However, developing a system which produces sound using physically-based principles in real time poses substantial difficulties. The foremost requirement is the presence of an efficient dynamics engine which informs the sound system of object collisions and the forces involved. For this work, we have developed a fast and robust rigid-body simulator, but many present day games meet this requirement. Given a dynamics simulator, the main challenge is to synthesis the sound efficiently enough to play in real time while taking only a small portion of the total running time, which is usually dominated by graphics and rigid-body simulation. Typically the sound system can only afford a few hundred CPU cycles per object per sound sample for many interactive applications.

In this course we present an approach which meets the interactive performance requirements outlined above, while ensuring high realism and fidelity of the sound produced. Given an object's geometry and a few material parameters, we construct a spring-mass model approximating the object's surface. We show that although a spring-mass system is a coarser approximation than FEM models used in prior



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 130: Applications

approaches [CD97, OCE01, OSG02], it is an adequate model to capture the small-scale surface vibrations that lead to the generation of sound in nature. We show how this formulation yields an analytical solution to the equation of motion for the surface of the object.

However, a naive implementation of such an approach can handle only a few (less than ten) sounding objects in real time. We also present several acceleration techniques. The increased computational efficiency is achieved by exploiting auditory perception, which ensures that the resulting degradation in perceived quality is minimal. In addition, the sound quality and the associated computational cost for each object is scaled dynamically in a priority-based scheme which guarantees that the total sound production meets stringent time constraints, while preserving the overall aural experience. Our approach has the following characteristics:

- It is based on a discretized physically-based representation that offers simplicity of formulation and ease of implementation;
- It makes no assumptions about the input mesh topology – surface meshes used for physics can be used directly for sound synthesis;
- It is capable of yielding both impact and rolling sounds naturally, without any special-case treatment;
- It enables rich environments consisting of numerous sounding objects, with insignificant difference in the overall audio quality.

We also use OpenAL and EAX™ to provide hardware-accelerated propagation modeling of the synthesized sounds on Creative Sound Blaster Audigy 2™ audio cards which easily produce spatial and environmental sound effects such as distance attenuation and room acoustics. To the best of our knowledge, with the possible exception of methods that rely on physical measurements, no prior work has been demonstrated to handle complex scenarios (e.g. see Figures 131 and 137) in real time.

29.1 Physically-based Models

The concept of modeling the surface vibrations of objects using discretized physical models in real time was first proposed by Florens and Cadoz [FC91], who used a system of masses and damped springs to model 3D shapes and developed the CORDIS-ANIMA system for physically-based sound synthesis. More recently, numerical integration with a finite element approach was proposed as a more accurate technique for modeling vibrations [CD97, OCE01]. These methods had the advantage that the simulation parameters corresponded directly to physically measurable quantities and the results were more accurate. The main drawback was the complexity of formulation and implementation and the low speed of the resulting simulation.

To remedy the performance issue of the above methods, van den Doel and Pai suggested [vdDP96, vdDP98] using the analytically computed vibrational modes of an object, instead of numerical integration, leading to considerable speedups and enabling real-time sound synthesis. But, since the PDEs governing the vibration of arbitrary shapes are very complicated, the proposed system could only handle simple systems, such as plates, for which the analytical solutions were known. To handle more complex systems which do not admit direct analytical solution, two approaches have been proposed in literature. The first approach [vdDKP01], uses physical measurements on a given shape to determine

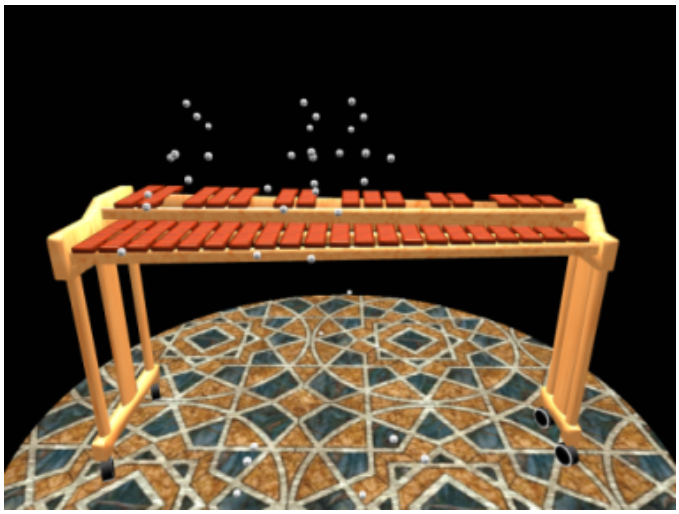


Figure 131: Numerous dice fall on a three-octave xylophone in close succession, playing out the song “The Entertainer” (see the video). Our algorithm is able to produce the corresponding musical tones at more than 500 FPS for this complex scene, with audio generation taking 10% of the total CPU time, on a 3.4GHz Pentium-4 Laptop with 1GB RAM.

its vibration modes and their dependence on the point of impact. Later, these modes may be appropriately mixed in a real-time application to generate realistic synthetic sounds. But, arbitrary 3D models have to be physically procured, in order to find their aural properties. In [OSG02], O’Brien et al. address this problem and propose a method for handling arbitrarily-shaped objects by discretizing them into tetrahedral volume elements. They show that the corresponding finite element equations can be solved analytically after suitable approximations. Consequently, they are able to model arbitrarily shaped objects and simulate realistic sounds for a few objects at interactive rates.

Our work shares some common themes with [OSG02]. However, we propose a simpler system of point-masses and damped springs for modeling the surface vibrations of the object and it also submits to an analytical solution in a similar fashion, while offering much greater simplicity of formulation and ease of implementation. Furthermore, the complexity of scenes demonstrated in [OSG02] is low, containing less than 10 sounding objects and the interactions captured are mainly due to impacts. As we will demonstrate in Section 5, our method extends to handling hundreds of objects in real time and is also capable of producing realistic *rolling* sounds in addition to impact sounds.

Often, immersive environments are both visually and aurally complex. The problem of scheduling multiple objects for sound synthesis was first addressed in [FBH97]. They exploited a model of imprecise computation proposed previously in [CLL87], and proposed a system in which the objects are iteratively assigned time quotas depending on the availability of resources and priorities of the objects. As described in Section 4, our approach to prioritization and time-quota assignment exploits properties specific to our sound-generation technique, and thus achieves better results using a much simpler scheme. Recently, van den Doel et al. [vdDKP04] proposed techniques to synthesize sounds in real time for scenes with a few sounding rigid bodies and numerous particles, by exploiting frequency masking. At runtime, they find emitted frequencies which are masked out by other neighboring frequencies with higher amplitude and do not mix the masked frequencies. We use a different perceptual observation presented in [SM95], which report that humans are incapable of distinguishing frequencies that are very close to each other. As we will discuss in Section 4, this can be used to prune out frequencies from an object’s frequency spectrum as a *pre-processing* step. Our technique leads to better performance and much lesser memory consumption at runtime while ensuring minimal loss in auditory quality.

29.2 Methodology

Sound is produced by surface vibrations of an elastic object under an external impulse. These vibrations disturb the surrounding medium to result in a pressure wave which travels outwards from the object. If the frequency of this pressure wave is within the range 20 to 22000 Hz, it is sensed by the ear to give us the subjective perception of sound. The most accurate method for modeling these surface vibrations is to directly apply classical mechanics to the problem, while treating the object as a continuous (as opposed to discrete) entity. This results in PDEs for which analytical solutions are not known for arbitrary shapes. Thus, the only avenue left is to make suitable discrete approximations of the problem to reduce the PDEs to ODEs, which are more amenable to analysis. In this section, we show how a spring-mass system corresponding to a physical object may be constructed to model its surface deformation and how it may be analyzed to extract the object’s modes of vibration. For ease of illustration, we assume a homogeneous object; inhomogeneous objects may be handled by a simple extension of the approach presented here. Further, we assume that the input object is in the form of a thin shell and is hollow inside. This assumption is motivated by practical concerns since most of the geometry today is modeled for rendering and is invariably only a surface representation with no guarantees on surface connectivity. If a volumetric model is available, the approach outlined in this paper applies with minor modifications. Figure 132 gives an overview of our approach.

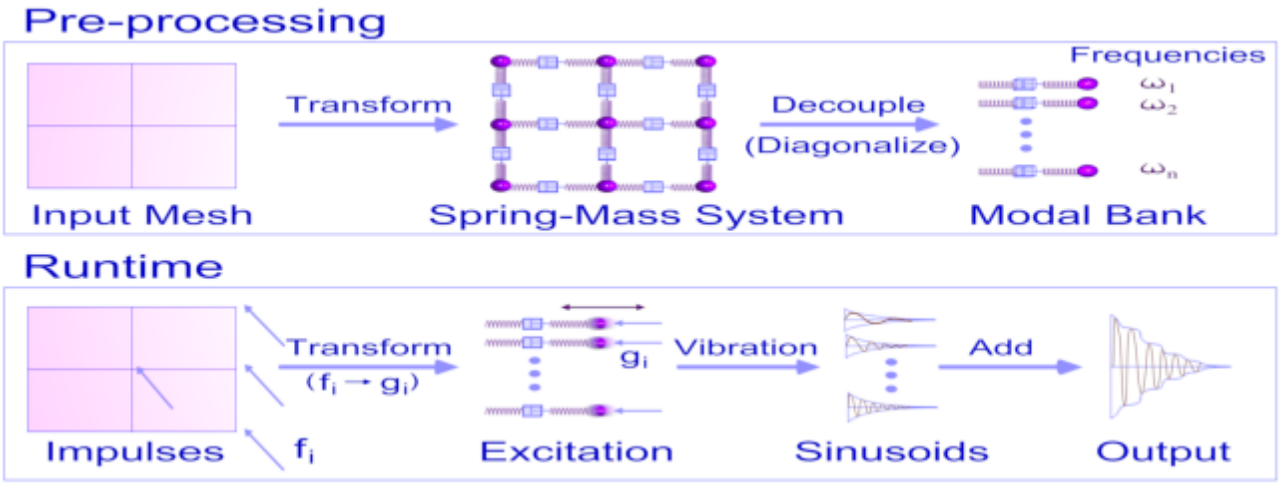


Figure 132: This diagram gives an overview of our approach. In the pre-processing step, each input surface mesh is converted to a spring-mass system by replacing the mesh vertices with point masses and the edges with springs, and the force matrices are diagonalized to yield its characteristic mode frequencies and damping parameters. At runtime, the rigid-body simulator reports the force impulses f_i on a collision event. These are transformed into the mode gains, g_i with which the corresponding modes are excited. These yield damped sinusoids which are suitably combined to yield the output sound signal.

29.2.1 Input Processing

Given an input mesh consisting of vertices and edges, we construct an equivalent spring-mass system by replacing the mesh vertices with point masses and the edges with damped springs. We now discuss how to assign the spring constants and masses based on the material properties of the object so that the discrete system closely approximates the physical object. The spring constant, k and the particle masses, m_i are given by:

$$\begin{aligned} k &= Yt \\ m_i &= \rho t a_i \end{aligned} \tag{28}$$

where Y is the Young's Modulus of elasticity for the material, t is the thickness of the object surface, ρ is the material density and a_i is the area "covered" by a particle, which is calculated by dividing the area of each mesh face equally amongst all its constituent vertices and summing all the face contributions for the vertex corresponding to the mass in consideration. Note that we did not discuss fixing the spring damping parameters above, which we will return to shortly.

29.2.2 Deformation and Vibration Modeling

Once the particle system has been constructed as above, we need to solve its equation of motion in order to generate the corresponding sound. Unfortunately, the resulting system of equations is still mathematically complex because the interaction forces between the masses are non-linear in their positions. However, by making the reasonable assumption that the deformation is small and linearizing about the rest positions, this problem can be cast in the form of a coupled linear system of ODEs:

$$M\frac{d^2r}{dt^2} + (\gamma M + \eta K)\frac{dr}{dt} + Kr = f \tag{29}$$

where M is the mass matrix, K is the elastic force matrix, γ and η are the fluid and viscoelastic damping constants for the material respectively. The matrix M is diagonal with entries on the diagonal corresponding to the particle masses, m_i. The elastic force matrix K is real symmetric, with entries relating two particles if and only if they are connected by a spring. The variable r is the displacement vector of the particles with respect to their rest position and f is the force vector. Intuitively, the terms in the above equation correspond to inertia, damping, elasticity and external force respectively. The specific form of damping used above, which expresses the overall damping matrix as a linear combination of K and M is known as Raleigh damping and works well in practice. For a system with N particles in three dimensional space, the dimensions of all the matrices above is $3N \times 3N$.

This formulation of the problem is well known and is similar to the one presented in [OSG02]. The main difference in our approach is that the force and inertia matrices are assembled from a spring-mass system which makes the formulation much simpler. The solution to Equation (29) can be obtained by diagonalizing K so that:

$$K = GDG^{-1} \tag{30}$$

where G is a real matrix consisting of the eigenvectors of K and D is a diagonal matrix containing the eigenvalues. For reasons we will explain later, we will henceforth call G the "gain matrix". Plugging the above expression for K into Equation (29) and multiplying by G^{-1}

throughout, we obtain:

$$G^{-1}M\frac{d^2r}{dt^2}+\left(\gamma G^{-1}M+\eta DG^{-1}\right)\frac{dr}{dt}+DG^{-1}r=f \tag{31}$$

Observing that since M is diagonal, $G^{-1}M=MG^{-1}$ and defining $z=G^{-1}r$ equation(31) may be expressed as:

$$M\frac{d^2z}{dt^2}+(\gamma M+\eta D)\frac{dz}{dt}+Dz=G^{-1}f \tag{32}$$

Since both M and D in the above equation are diagonal, Equation (29) has been decoupled into a set of unrelated differential equations in the variables z_i, which correspond to individual modes of vibration of the object. The equation for each mode is the standard equation of a damped oscillator and has the following solution for the i'th mode:

$$\begin{aligned} z_i(t) &= c_i e^{\omega_i^+ t}+\overline{c_i}e^{\omega_i^- t} \\ \omega_i^{\pm} &= \frac{-(\gamma\lambda_i+\eta)\pm\sqrt{(\gamma\lambda_i+\eta)^2-4\lambda_i}}{2} \end{aligned} \tag{33}$$

where the constant c_i, called the gain for the mode, is found by considering the impulses applied as we will discuss shortly. We use $\overline{c_i}$ to denote the complex conjugate of c_i. The constant λ_i is the i'th eigenvalue in the diagonal matrix, D. The real part of $\omega_i^{\pm}$ gives the damping coefficient for the mode, while the imaginary part, if any, gives the angular frequency of the mode.

29.2.3 Handling Impulsive Forces

Once an input mesh has been processed as above and the corresponding modes extracted as outlined in Equations (29)-(33), we have all the information needed regarding the aural properties of the object. The sound produced by an object is governed by the magnitude and location of impulsive force on its surface. We model short-duration impulsive contacts by dirac-delta functions. Given an impulse vector f containing the impulses applied to each vertex of an object, we compute the transformed impulse, $g=G^{-1}f$ in order to evaluate the right-hand side of Equation (32). Once this is done, the equation for the i'th mode is given by:

$$m_i\frac{d^2z_i}{dt^2}+(\gamma m_i+\eta\lambda_i)\frac{dz_i}{dt}+\lambda_i z_i=g_i\delta(t-t_0) \tag{34}$$

where t_0 is the time of collision and $\delta()$ is the dirac delta function. Integrating the above equation from a time just before t_0 to a time just after t_0 and noting that $\int_{t_0}^{t_0^+}\delta(t-t_0)dt=1$, we obtain:

$$m_i\Delta\left(\frac{dz_i}{dt}\right)+(\gamma m_i+\eta\lambda_i)\Delta z_i+z_i\Delta t=g_i \tag{35}$$

Assuming that Δt is very small, and using the fact that the deformation is small compared to the change in velocities, we can neglect the last two terms on the left-hand side to obtain:

$$\Delta\left(\frac{dz_i}{dt}\right)=\frac{g_i}{m_i} \tag{36}$$

The above gives a very simple rule which relates the change in the time derivative of the mode to the transformed impulse. Referring to Equation (33) and requiring that z_i should stay the same just before and after the collision while $\frac{dz_i}{dt}$ should increase as in Equation (36), the update rule for the mode gain c_i can be shown to be:

$$c_i \leftarrow c_i+\frac{g_i}{m_i\left(\omega_i^+-\omega_i^-\right)e^{\omega_i^+ t_0}} \quad . \tag{37}$$

Initially, c_i is set to 0 for all modes.

29.3 Exploiting Auditory Perception

In this section, we describe how the mathematical formulation presented in the previous section is utilized to efficiently generate sound in real time. First, we describe a naive implementation and then discuss techniques to increase its efficiency.

Assume that there exists a rigid-body simulator which can handle all the dynamics. During a collision event, the sound system is informed of the object that undergoes the impact and the magnitude and location of impact. This impact is processed as described in Section 29.2.3 to result in the gains for the different modes of vibration of the object, where the gain for the i'th mode being c_i. The equation for a mode from the time of collision onwards is given by (33). The amplitude contribution of a mode at any moment is proportional[8] to its *velocity* (and not

[8]The constant of proportionality is determined based on the geometry of the object and takes the fact into account that vibrations in the direction of the surface normal contribute more to the resulting pressure wave than vibrations perpendicular to the normal. We do not describe it in detail here as it is not critical to the approach being presented.

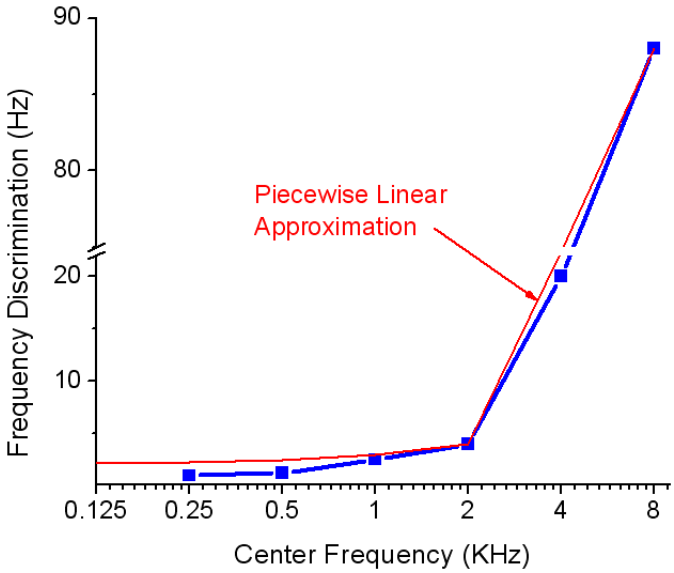


Figure 133: This plot shows frequency discrimination in humans as a function of the center frequency. Note that the human capacity to discriminate between frequencies degrades considerably for frequencies in the range 2-22 KHz, which forms a bulk of the human audible range. We use this fact to guarantee that no more than 1000 modes need to be mixed for any object in the worst case, *irrespective* of its geometric complexity. In most cases the actual number is much smaller, in the range of a few hundreds. The red curve shows the piecewise linear approximation of this curve that we use.

position). This is because the pressure contribution of a particle is determined by its velocity and the mode velocities are linearly related to the physical velocities of the particles. The mode velocity is found by taking a differential of Equation (33) with respect to time:

$$v_i = \frac{dz_i}{dt} = c_i \omega_i^+ e^{\omega_i^+ t} + \overline{c_i} \omega_i^- e^{\omega_i^- t} \tag{38}$$

For generating each audio sample, we need to evaluate the above equation for all vibration modes of the object, which is quite inefficient. As mentioned in [OSG02], the simple observation that $e^{i\omega(t+\Delta t)} = e^{i\omega t} e^{i\omega \Delta t}$ offers some gain in performance since generating a new audio sample just requires a single complex multiply with the previous value. However, the efficiency is still not sufficient to handle a large number of objects in real time. We may estimate the running time of the system as follows: A simple spring-mass system with N particles has $3N$ modes, and the above operation needs to be repeated for each mode for each audio sample. Assuming a sampling rate of 44100 Hz, the number of floating-point operations (FLOPS) needed for this calculation for generating audio samples worth t seconds is:

$$T = 3N \times 4 \times 44100t \; FLOPS \; . \tag{39}$$

Considering that the typical value of N is about 5000 or higher, producing sound worth 1 second would take 2646 MFLOPS. Since today's fastest processors operate at a few thousand MFLOPS [Don05], the above processing would take about a second. Given that this estimated amount of time is for just one object and a typical scene may contain many such objects, such an approach is clearly not fast enough for interactive applications. Furthermore, for many real-time environments such as games and virtual environments, only a very small fraction of the actual time can be allocated for sound production. Thus, in the rest of this section, we will discuss techniques to increase the efficiency of the proposed base system to enhance its capability in handling scenarios with a large number of sounding objects at interactive rates.

From Equation (39), it is clear that the running time is proportional to the number of modes being mixed and the number of objects. Next, we present acceleration techniques for sound simulation by reducing the number of modes per object: "Mode Compression" and "Mode Truncation", and by scaling the audio quality of different objects dynamically with little degradation in perceived sound quality.

29.3.1 Mode Compression

Humans have a limited range of frequency perception, ranging from 20 to 22000 Hz. It immediately follows that modes with frequencies lying outside this range can be clipped out and need not be mixed. However, there is another important fact which can lead to large reductions in the number of modes to be mixed. A perceptual study described in [SM95] shows that humans have a limited capacity to discriminate between nearby frequencies. Note that this is different from frequency masking [ZF90] in which one of two *simultaneously* played frequencies masks out the other. Rather, this result reports that even if two "close enough" frequencies are played in succession, the listener is unable to tell whether they were two different frequencies or the same frequency played out twice. The authors call the length of the interval of frequencies around a center frequency which sound the same, the "Difference Limens to Change" (DLC). Figure 133 shows a plot of the DLC against center frequencies ranging from .25 to 8 KHz. Interestingly, the DLC shows a large variation over the audible frequency range, getting very

large as the center frequency goes beyond 2 KHz. Even at 2 KHz, the DLC is more than 1 Hz. That is, a human subject cannot tell apart 1999 Hz from 2000 Hz.

We use the above fact to drastically reduce the number of modes that are mixed for an object. We linearly approximate the DLC curve with a piecewise linear curve shown as the red line in Figure 133. The approximation has two segments: one from 20 Hz to 2 KHz and another from 2 KHz to 22 KHz. As we show in the figure we overestimate the DLC slightly. This increases the performance further and we have observed minimal loss in quality in all the cases we have tested. The main idea behind our compression scheme is to group together all the modes with perceptually indistinguishable frequencies. It can be easily shown that if the above mentioned linear approximation to the DLC curve is used and indistinguishable modes clustered at the corresponding frequency centers, the maximum number of modes that need to be mixed is less than 1000. It is important to note that this is just the worst case scenario and it happens only when the frequency spectrum of the object consists of all frequencies from 20 to 22,000 Hz, which is very rare. For most objects, the frequency spectrum is discrete and consequently, the number of modes after mode compression is much smaller than 1000, typically in the range of a few hundreds.

We now describe the details of our technique. Recall the gain matrix from Equation (30), G. The gain matrix has a very simple physical interpretation: Rows of the matrix correspond to vertices of the object and columns correspond to the different modes of vibration (with their corresponding frequencies). Each row of G lists the gains for the various modes of vibration of the object, when a unit impulse is applied on the corresponding vertex. It is clear from the above discussion that all the mode gains within a row of G which correspond to modes with close frequencies need to be clustered together. This is achieved by replacing the gain entries for all such modes by a single entry with gain equal to the sum of the constituent gains. Since a mode corresponds to a whole column, this reduces to summing together columns element-wise based on their frequencies. The complete procedure is as follows:

- Sort the columns of G with the corresponding mode frequencies as the key. [9]
- Traverse the modes in increasing order of frequency. Estimate the DLC, Δ at the current frequency using the piecewise linear curve shown in Figure 133. If the current frequency and next frequency are within Δ of each other the two mode frequencies are indistinguishable, replace the two columns by their element-wise sum.

Below, we enumerate the main advantages of this scheme:

1. The running time is constant instead of linear in the number of vertices in the object. For example, if the input mesh is complex with 5,000 vertices, the number of modes mixed is bounded by 1000 instead of the earlier $3N = 15,000$ which is a substantial performance gain.
2. Since this scheme requires just the frequencies of the different modes, the whole processing can be done as a pre-process without requiring any extra runtime CPU cycles.
3. From the above mentioned procedure, it is clear that the number of columns in the matrix G, which is the same as the number of modes, is now bounded by 1000 instead of the earlier value of $3N$. Since this matrix needs to be present in memory at runtime for transforming impulses to mode gains, its memory consumption is an important issue. Using this technique, for an object with 5000 vertices, the memory requirement has been reduced from 225 MB to less than 15 MB, by more than a factor of 15.
4. Most objects have a discrete frequency spectrum with possibly many degenerate frequencies. Due to numerical inaccuracies while diagonalizing the elastic force matrix and the approximations introduced by the spring-mass discretization, these degenerate frequencies may appear as spurious distinct modes with near-equal frequencies. Obviously, it is wasteful to treat these as distinct modes. It is our observation that most of the times these modes' frequencies are close enough so that they are naturally summed together in this scheme.

29.3.2 Mode Truncation

The sound of a typical object on being struck consists of a transient response composed of a blend of high frequencies, followed by a set of lower frequencies with low amplitude. The transient attack is essential to the quality of sound as it is perceived as the characteristic "timbre" of the object. The idea behind mode truncation is to stop mixing a mode as soon as its contribution to the total sound falls below a certain preset threshold, τ. Since mode truncation preserves the initial transient response of the object when τ is suitably set, the resulting degradation in quality is minimal. Figure 134 shows a plot of the number of active modes with respect to time for a xylophone bar struck in the middle for two different values of τ: .01 and 2. These values are normalized with respect to the maximum sample value which is 65536 for 16-bit audio. The first case with $\tau = .01$ performs essentially no truncation, only deactivating those modes which have near-zero amplitude. Note that with $\tau = 2$ the number of modes mixed is reduced by more than 30%. Also, the number of active modes floors off much earlier (.2 secs compared to .6 secs). It is important to note that this results in little perceptible loss in quality.

The details of the technique are as follows: Assume that an object has just undergone a collision and the resulting mode gains c_i have been calculated as given by Equation (37). From this time onwards until the object undergoes another collision, Equation (38) gives a closed-form expression for the mode's contribution to the sound of the object. This can be used to predict exactly when the mode's contribution drops below the threshold τ. The required "cutoff time", t_i^c is such that for all times $t > t_i^c$:

$$c_i \omega_i^+ e^{\omega_i^+ t} + \overline{c_i} \omega_i^- e^{\omega_i^- t} < \tau \tag{40}$$

[9] This step is usually not needed as most linear algebra packages output the eigenvector matrix sorted on the eigenvalues

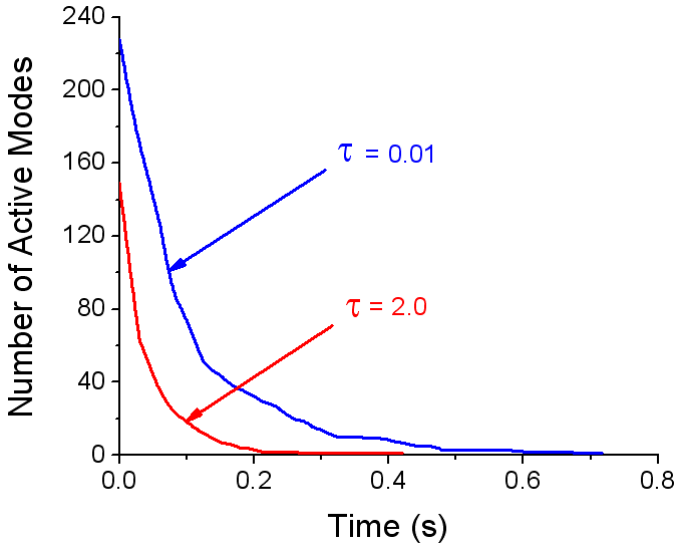


Figure 134: This graph shows the number of modes mixed vs time, for a xylophone bar just after it is struck in the middle. τ is the mode truncation threshold. A higher value of τ leads to more aggressive truncation of modes with low amplitude, leading to savings in terms of the number of modes mixed. In this case, $\tau = 2.0$ results in about 30% gain in efficiency over $\tau = 0.01$ which only truncates modes with near-zero amplitude. The sound quality for both the cases is nearly identical.

Using the fact that for any two complex numbers x and y, $|x+y| \leq |x| + |y|$ it can be shown that,

$$t_i^c \leq \frac{1}{-Re(\omega_i^+)} \ln\left(\frac{2\,|c_i|\,|\omega_i^+|}{\tau}\right) \tag{41}$$

Using the above inequality, the cutoff times are calculated for all the modes just after a collision happens. While generating the sound samples from a mode, only one floating point comparison is needed to test if the current time exceeds the cutoff time for the mode. In case it does, the mode's contribution lies below τ and consequently, it is not evaluated.

29.3.3 Quality Scaling

The two techniques discussed above are aimed at increasing the efficiency of sound synthesis for a single object. However, when the number of sounding objects in a scene grows beyond a few tens, this approach is not efficient enough to work in real time and it is not possible to output the sound for all the objects at the highest quality. It is critical in most interactive applications that the sound system have a graceful way of varying quality in response to variable time constraints. We achieve this flexibility by scaling the sound quality for the objects. The sound quality of an object is changed by controlling the number of modes being mixed for synthesizing its sound. In most cases of scenes with many sounding objects, the user's attention is on the objects in the "foreground", that is, the objects which contribute the most to the total sound in terms of amplitude. Therefore, if it is ensured that the foreground sounds are mixed at high quality while the background sounds are mixed at a relatively lower quality, the resulting degradation in perceived aural quality should be reduced.

We use a simple scheme to ensure higher quality for the foreground sounds. At the end of each video frame, we store the sum of the vibrational amplitudes of all modes for each object, which serve to determine the object's priority. At the next video frame, all objects are sorted in decreasing order based on their priority and the total time-quota for sound-generation divided among the objects as a linearly decreasing ramp with a preset slope, S. After this, all objects are processed in their priority order. For each object, its quality is first scaled so that it can finish within its assigned time-quota and then the required modes are mixed for the given time period. If an object finishes before its time-quota has expired, the surplus is consumed greedily by the next higher priority object. The slope, S of the ramp decides the degree to which the foreground sound quality is favored over a degradation in background sound quality. The case with $S = 0$ corresponds to no priority scheduling at all, with the time-quota being divided equally among all objects. The converse case with $S = \infty$ corresponds to greedy consumption of the time-quota. That is, the whole time-quota is assigned to the highest priority object. After the object is done, the remaining time, if any, is assigned to the next highest priority object and so on.

To illustrate how all the techniques described above are integrated, we present a summary of our approach.

Pre-processing

- Construct a spring-mass system corresponding to the input mesh. (Section 29.2.1)

- Process the spring-mass system to extract the gain matrix, G and the (complex) angular frequencies of the object's modes of vibration: ω_i^+ and ω_i^-. (Section 29.2.2, Eqns. (30) and (33))
- **Mode Compression:**
  Aggregate columns of G based on frequencies of the corresponding modes, as described in Section 29.3.1.
- Store the resulting gain matrix along with the (complex) constants ω_i^+ and ω_i^- for modes correspond to the columns of G after compression. Note that ω_i^- need not be stored in case ω_i^+ has a non-zero imaginary part since in that case $\omega_i^- = \overline{\omega_i^+}$.

Runtime Processing

- Load the gain matrix and mode data for each object.
- Begin simulation loop:
  1. Run rigid-body simulation
  2. For each object, O:
     - **Collision Handling:**
       If the rigid-body simulator reports that O undergoes a collision event, update its gain coefficients as per Equation (37) using the collision impulse and its location. (Section 29.2.3)
     - **Mode Truncation:**
       Compute cutoff times t_j^c for each mode based on the mode truncation threshold, τ. (Section 29.3.2, Equation (41))
  3. **Quality Scaling:**
     Sort objects based on amplitude contribution, assign time-quotas and compute the number of modes to be mixed for each object. (Section 29.3.3)
  4. **Sound Synthesis:**
     For each time-step at time t and for each object, O:
     - Consider all modes permitted by the current quality setting which satisfy $t_j^c > t$. Sample and summate all these modes as described at the beginning of this section. This is O's contribution to the sound.
     - Output the sum of all objects' sample contribution as the sound sample for time t.

  End simulation loop

29.3.4 Position Dependent Sounds

We present results to demonstrate the efficiency and realism achievable with our approach.

29.4 Rigid Body Simulation

We have implemented the algorithm and acceleration techniques presented in this paper using C++ and OpenGL. Our rigid-body simulator extends the technique presented by Guendelman et al. [GBF03] to incorporate DEEP [KLM02a] for fast and more accurate penetration depth estimation, instead of sample-based estimation using distance fields. It also uses a more complex friction model presented by Mirtich and Canny [MC95], which results in more robust contact resolution.

As discussed earlier, the main advantage of using physically-based sounds over recorded audio is the ability to capture effects such as the magnitude of impacts between objects and more importantly, the subtle shift in sounds on striking an object at different points. Figure 135 shows a scene with a metallic cylinder tossed onto a wooden table. Both the table and cylinder are sounding. The figure contrasts two cases: the first case, shown on the left, depicts the cylinder striking the table near the middle and rolling off, while in the second case it strikes the table near the edge. We discuss the rolling sound in the next subsection, and will discuss the impact sound here. Since the table-top is in the form of a plate, we would expect that striking it on the edge would transfer a larger fraction of the impulse to higher frequencies, while striking it in the middle should transfer most part of the impulse to the fundamental mode of vibration, leading to a deeper sound. To verify this, we plotted the frequency spectra for the two cases just after the cylinder makes first impact with the table. The corresponding plots for the two cases are shown in the lower part of the figure. The case on the left shows a marked peak near the fundamental mode while the peak is completely missing in the second case. Conversely, the second case shows many peaks at higher frequencies which are missing in the first one. This difference clearly demonstrates that the sound for the two cases is markedly different, with the same qualitative characteristics as expected. Another important point to note is that this technique does not require the meshes to be highly tessellated to capture these effects. The table consists of just 600 vertices and the cylinder 128 vertices.

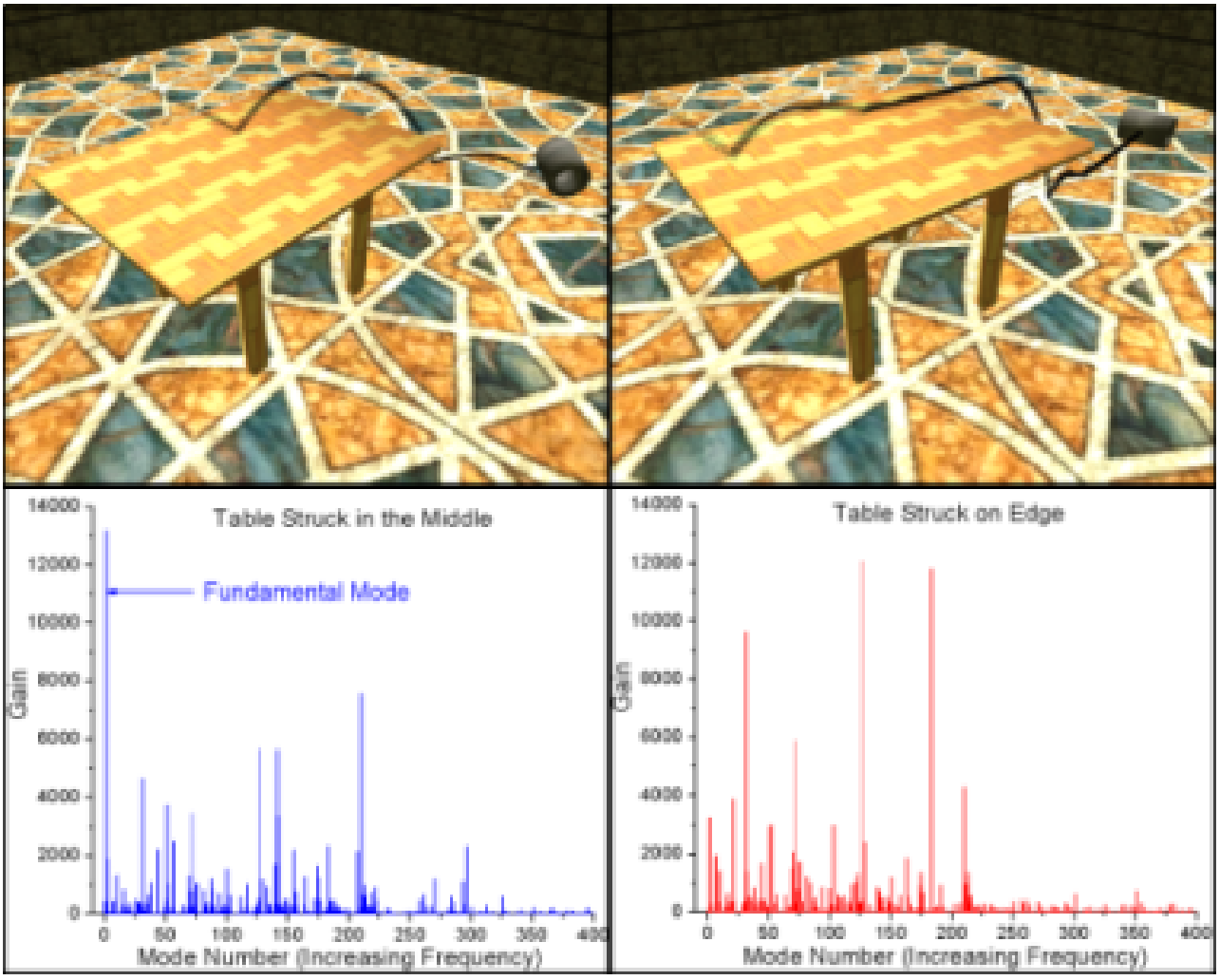


Figure 135: A metallic cylinder falls onto a wooden table, in the middle (left) and on the edge (right) and rolls off. The bottom part shows the corresponding frequency spectra for the two cases. Note that for the case on the left, most of the impulse is transferred to the low frequency fundamental mode while for the case on the right, the impulse is mostly transferred to higher frequency modes.

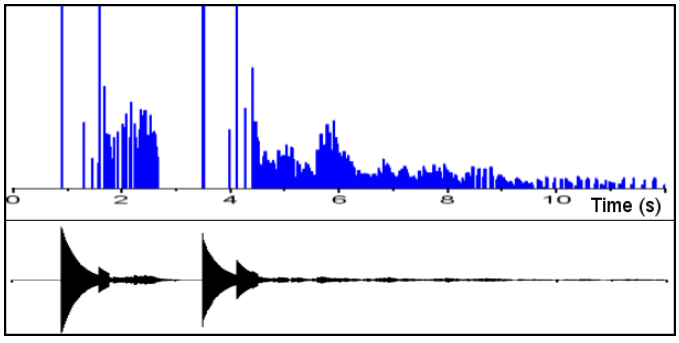


Figure 136: A plot of the impulses on a cylinder versus time for the scene shown on the right in Figure 135 and the corresponding audio samples. The peaks correspond to impacts while the numerous low-amplitude impulses correspond to rolling forces.

29.4.1 Rolling Sounds

In addition to handling impact sounds, we are able to simulate realistic rolling sounds without requiring any special-case treatment for sound synthesis. This is in part made possible because of the rigid-body simulator we have developed, which is able to handle contacts in a more graceful manner than most impulse-based simulators. Figure 136 shows the impulses on the cylinder and the corresponding audio for the case shown in the right side of Figure 135. The cylinder rolls on the table after impact, falls to the ground and rolls on the floor for sometime. The initial rolling sound, when the cylinder is on the table, has a much richer quality. The sound of the table as the cylinder rolls over it conveys a sense of the cylinder's heaviness, which is only partly conveyed by the sound of the impact. The cylinder, although uniformly tessellated, is very coarse, with only 32 circumferential subdivisions. Figure 136 shows the impulses applied on the cylinder against time. The peaks correspond to impacts: when the cylinder falls on the table, and when it falls to the ground from the table. Note that the audio waveform shows the corresponding peaks correctly. The period of time stretching from 6 to 8 seconds consists of the cylinder rolling on the floor and is characterized by many closely-spaced small-magnitude impulses on the cylinder as it strikes the floor again and again due to its tessellation. To test how important the periodicity of these impulses was for the realism of rolling sounds, we found the mean and standard deviation of the interval between these impulses from the data presented in Figure 136. The mean time between the impulses was 17 ms with a standard deviation of 10 ms. The fact that the standard deviation is more than 50% of the mean demonstrates that the impulses show very little periodicity. This suggests that the periodicity of collisions is not critical for the perceived realism of rolling sounds.

29.4.2 Efficiency

We are able to do audio simulation for complex scenes in real time using our approach. Figure 137 shows a scene with 100 metallic rings falling simultaneously onto a wooden table and undergoing elastic collision. All the rings and the table are sounding. Each ring is treated as a separate object with separate aural properties. The rings consist of 200 vertices each. Figure 138 shows the audio FPS[10] for this simulation against time for the one second interval during which almost *all* the collisions take place. The application frame rate is 100 FPS. Note that this is not the raw data but a moving average so that the short-range fluctuations are absorbed. The plot on the bottom is the base timing without using any of the acceleration techniques described in Section 4. The audio in this case is very choppy since the audio generation is not able to keep up with the speed of rendering and rigid-body simulation. With mode truncation and mode compression, the performance shows significant improvement. However, after initially starting at about 200 FPS, the frame rate drops in the latter part where the maximum number of collisions happen. With quality scaling in addition to mode compression and truncation (shown by the top-most curve), the frame rate exhibits no such drop, continuing to be around 200 FPS. This is because quality scaling gives priority to sound generation for those rings which just underwent collision, while lowering the quality for other rings which may have collided earlier and are contributing less to the overall sound. This illustrates the importance of quality scaling for scenarios with multiple collisions. It is important to note that although this example sufficiently demonstrates the capability of the system to maintain steady frame rates, it is improbable in a real application, since there are about 100 collisions within a second. This is the reason why the CPU utilization is high (50%). A more common scenario would be as shown in Figure 131, which has a much lower CPU utilization (10%).

To illustrate the realistic sounds achievable with our approach, we designed a three-octave xylophone shown in Figure 131. The image shows many dice falling onto the keys of the xylophone to produce the corresponding musical notes. The audio simulation for this scene runs in the range of 500-700 FPS, depending on the frequency of collisions. The dice have been scripted to fall onto the xylophone keys at precise moments in time to play out any set of musical notes. Because of the efficiency of the sound generation process, the overall system is easily able to maintain a steady frame rate of 100 FPS. Also, there are situations in which many dice fall on different keys within a few milliseconds of each other, but the sound quality exhibits no perceptible degradation. Although we have not tuned the xylophone keys to match the exact frequency spectrum of a real xylophone, the resulting sound is realistic and captures the essential timbre of the instrument. The material parameters for the xylophone were taken from [CD97].

We have presented a physically-based sound synthesis algorithm with several acceleration techniques for rendering a large-scale scene consisting of hundreds of interacting objects in real time, with little loss in perceived sound quality. This approach requires no special mesh structure, is simple to implement, and further takes advantage of existing hardware acceleration. We plan to extend this framework to auditory display of sliding sounds, explosion noises, breaking sounds, and other more complex audio effects that are difficult to simulate at interactive rates.

Part IX

State of the Art of Haptic Interaction

30 Fundamentals of Haptic Rendering

For a long time, human beings have dreamed of a virtual world where it is possible to interact with synthetic entities as if they were real. To date, the advances in computer graphics allow us to *see* virtual objects and avatars, to *hear* them, 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.

[10] An audio frame is defined as the amount of sound data sufficient to play for a duration equal to the duration of one video frame.

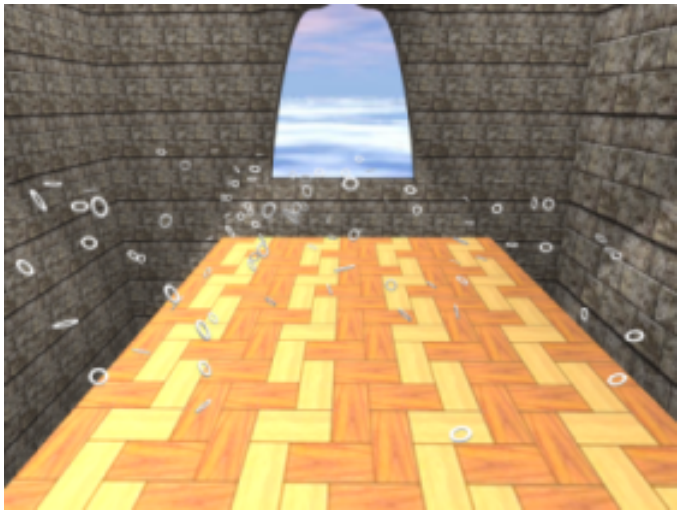


Figure 137: More than 100 metallic rings fall onto a wooden table. All the rings and the table are sounding. The audio simulation runs at more than 200 FPS, the application frame rate being 100 FPS. Quality Scaling ensures that the perceived sound quality does not degrade, while ensuring steady frame rates (See Figure 138)

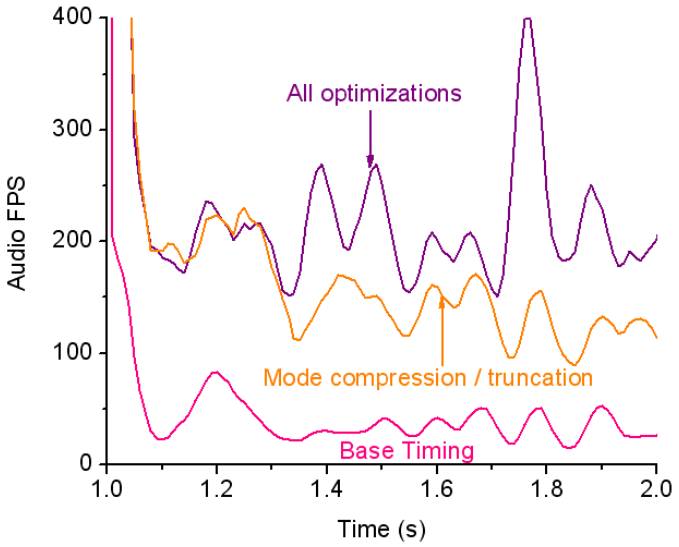


Figure 138: This graph shows the audio simulation FPS for the scene shown in Figure 137 from time 1s to 2s, during which almost all the collisions take place. The bottom-most plot shows the FPS for an implementation using none of the acceleration techniques. The top-most curve shows the FPS with mode compression, mode truncation and quality scaling. Note how the FPS stays near 200 even when the other two curves dip due to numerous collisions during 1.5-2.0s.

In §30, §33, §35, and §36 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 §33, 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 §35 we describe a multi-resolution collision detection algorithm based on the sensation-preserving simplification of complex geometry, and in §36 we describe techniques for perceptually-driven haptic texture rendering.

30.1 Introduction

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

30.1.1 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 139 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.

Figure 139: **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.*

30.1.2 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 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 commercialized and has become one of the most commonly used force-feedback devices.

But in the late '90s, research in haptic rendering revived one of the problems that first inspired virtual force feedback: 6-DoF haptic rendering or, in other words, grasping of a virtual object and synthesis of kinesthetic feedback of the interaction between this object and its environment.

Research in the field of haptics in the last 35 years has covered many more areas than what we have summarized here. [Bur96] gives a general survey of the field of haptics and [MHS02] discuss current research topics.

30.1.3 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 behaviors 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 generalized 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 summaries, 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.

30.2 The Challenges

Haptic rendering is in essence an interactive activity, and its realization 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.

30.2.1 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.

Essentially for 3DoF haptic rendering is limited to applications where point-object interaction is sufficient. For example haptic visualization of data, painting and sculpting, and some medical applications. Figure 140 illustrates the basic concept. First it is necessary to check if the manipulated point penetrates an object. Then determine the closest point on the surface. Finally a penalty-based force is exerted.

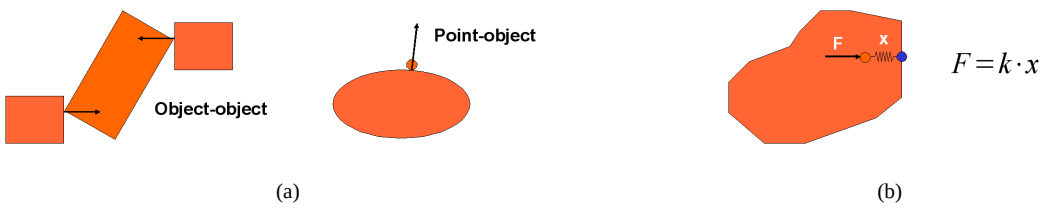
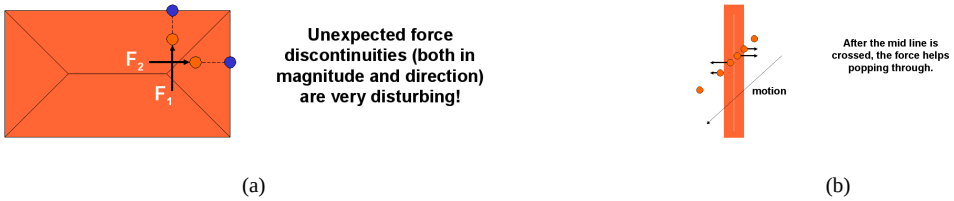


Figure 140: Introduction to haptic rendering

However, there are some problems listed below and illustrated in Figure 141.

- Force discontinuities arise when crossing boundaries of internal Voronoi cells.

- Pop-through thin objects.



(a) (b)

Figure 141: Problems for 3DoF haptic rendering

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 §33, 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 §30.3, 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.

30.2.2 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 behavior under large stiffness values. System instability is directly perceived by the user in the form of disturbing oscillations. A key factor for achieving a high dynamic range of impedances (i.e., stiff virtual walls) while ensuring stable rendering is the computation of feedback forces at a high update rate [CS94, CB94]. Brooks et al. [BOYBK90] reported that, in the rendering of textured surfaces, users were able to perceive performance differences at force update rates between 500Hz and 1kHz.

A more detailed description of the stability issues involved in the synthesis of force feedback, and a description of related work, are given in §30.3. Although here we have focused on impedance-type haptic devices, similar conclusions can be drawn for admittance-type devices (See [AH98a] and §30.3).

30.2.3 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 §28, 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.

30.3 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 §30.2.2, 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.

30.3.1 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 §30.2.1, 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.

30.3.2 Stable Rendering of Virtual Walls

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

Ouh-Young [OY90] created a discrete model of the Argonne ARM and the human arm and analyzed the influence of force update rate on the stability and responsiveness of the system. Minsky, Brooks, et al. [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 analyzed the influence of position sensors and quantization, and concluded that sensor resolution must be maximized and the velocity signal must be filtered.

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

30.3.3 Passivity and Virtual Coupling

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

Human Sensing and Control Bandwidths

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

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

Passivity of Virtual Walls

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

$$b > \frac{KT}{2} + B. \tag{42}$$

Stability of Non-Linear Virtual Environments

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

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

30.3.4 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.

31 Overview of State-of-Art Techniques

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

This reduced complexity has made 3-DoF haptic rendering an attractive solution for many applications with virtual haptic feedback, such as: sculpting and deformation [DQ*99, GEL00, MQW01], painting [JTK*99, GEL00, BSLM01, FOL02, AWD*04], volume visualization [AS96], nanomanipulation [TRC*93], and training for diverse surgical operations [KKH*97, GSM*97]. In each of these applications, the interaction between the subject and the virtual objects is sufficiently captured by a point-surface contact model.

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.

31.1 Constraint-based (God-objects, Virtual-proxy) 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 (Haptic Interface Point HIP) as a constrained optimization 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 practice, 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. This method is summarized in Figure 142.

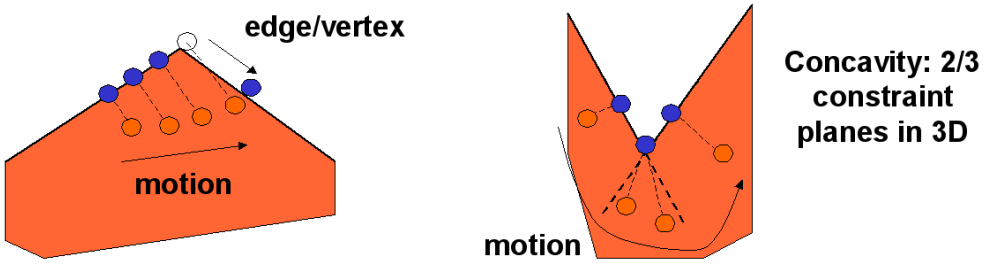
Ruspini et al. [RKK97] followed a similar approach to Zilles and Salisbury, which they called *virtual proxy* (VP). They modeled the contact point as a sphere of small radius and solved the optimization problem in the configuration space. As opposed to the *god object* method, they also formulated the 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 optimization problem and found a subgoal position. They repeated this process until the subgoal position was fully constrained. Ruspini et al. realized that the quadratic optimization 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 behavior to the contact point). This method is summarized in Figure 143.

We briefly outline the virtual proxy quadratic programming approach:

- The constraint planes define an open convex region (bounded by the plane at infinity).
- The function to minimize is the distance from the haptic device (HIP) to the new subgoal (VP_{i+1}): $C = ||VP_{i+1} - HIP(t+\Delta t)||$
- The translation from the current location to the new subgoal cannot intersect the constraint planes. Define linear constraints based on the normals of the planes. $N_i \cdot (VP_{i+1} - VP_i) \geq 0$

The force output proportional derivative (PD) control produces a force that will try to keep the virtual proxy and the HIP at the same position. The basic idea is summarized in Figure 144.

Force shading by Ruspini et al. [RKK97] shown in Figure 145 works by modifying the position of the virtual proxy. A subgoal (HIP') is computed changing the normal of the plane and this subgoal replaces the HIP. Finally the actual virtual proxy is computed.



(a)

$$A_i \cdot x + B_i \cdot y + C_i \cdot z + D_i = 0$$

$$E = \frac{1}{2} \cdot k \cdot \left((x - x_{HIP})^2 + (y - y_{HIP})^2 + (z - z_{HIP})^2 \right)$$

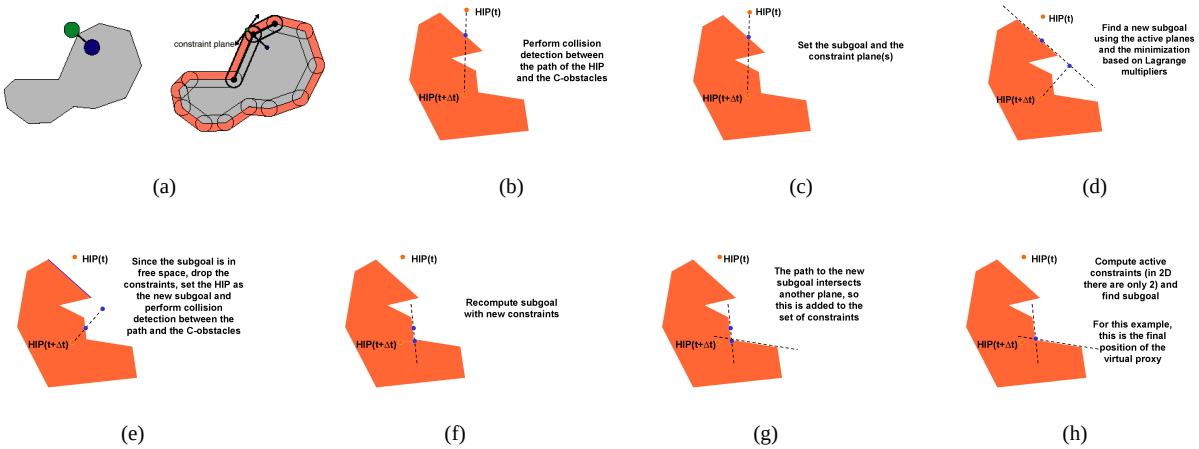
$$C = \frac{1}{2} \cdot k \cdot \left((x - x_{HIP})^2 + (y - y_{HIP})^2 + (z - z_{HIP})^2 \right) + \\ + \sum_{i=1}^{3} \lambda_i \cdot (A_i \cdot x + B_i \cdot y + C_i \cdot z + D_i)$$

(b)

$$\begin{pmatrix} 1 & 0 & 0 & A_1 & A_2 & A_3 \\ 0 & 1 & 0 & B_1 & B_2 & B_3 \\ 0 & 0 & 1 & C_1 & C_2 & C_3 \\ A_1 & B_1 & C_1 & 0 & 0 & 0 \\ A_2 & B_2 & C_2 & 0 & 0 & 0 \\ A_3 & B_3 & C_3 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix} = \begin{pmatrix} x_{HIP} \\ y_{HIP} \\ z_{HIP} \\ D_1 \\ D_2 \\ D_3 \end{pmatrix}$$

(c)

Figure 142: 3DoF haptics: God-object



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 143: 3DoF haptics: Virtual proxy

32 H-COLLIDE: Scalable Collision Detection for Three-Degree-of-Freedom (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

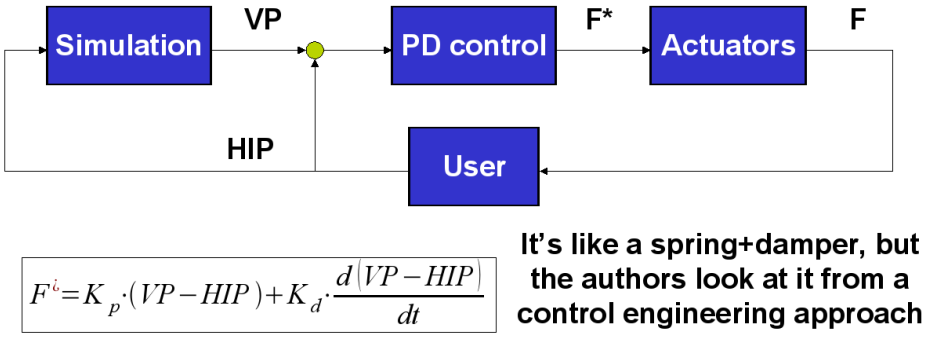


Figure 144: Proportional- derivative control

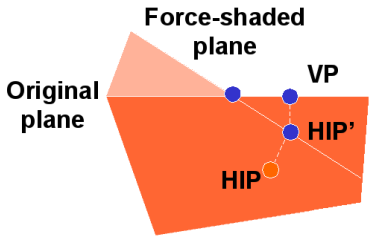


Figure 145: Force shading

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 optimized 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 pre-processing 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. The H-Collide system architecture is shown in Figure 146

We give an overview of the process illustrated in Figure 147. Firstly in an offline process a hybrid representation is pre-computed, consisting of uniform grids and each contains an OBBTree. Second, in a runtime process the "contact region" is identified using uniform spatial partitioning (implemented with hash table). The exact contact points are located by querying and traversing the OBBTrees. Frame-to-frame coherence is exploited by caching the last "witness" and finally the projected surface contact point found.

Figure 148 shows H-Collide in use to explore nano surfaces as part of the UNC Nanomanipulator project.

32.1 Application to Multi-resolution Editing and 3D Painting

Figure 149 shows some applications which have been developed using H-Collide as part of the inTouch system for direct haptic interaction. The system allows both interactive multi-resolution editing and painting of polygonal meshes using haptic brushes and similar tools.

33 Techniques for 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. §35 and §36 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.

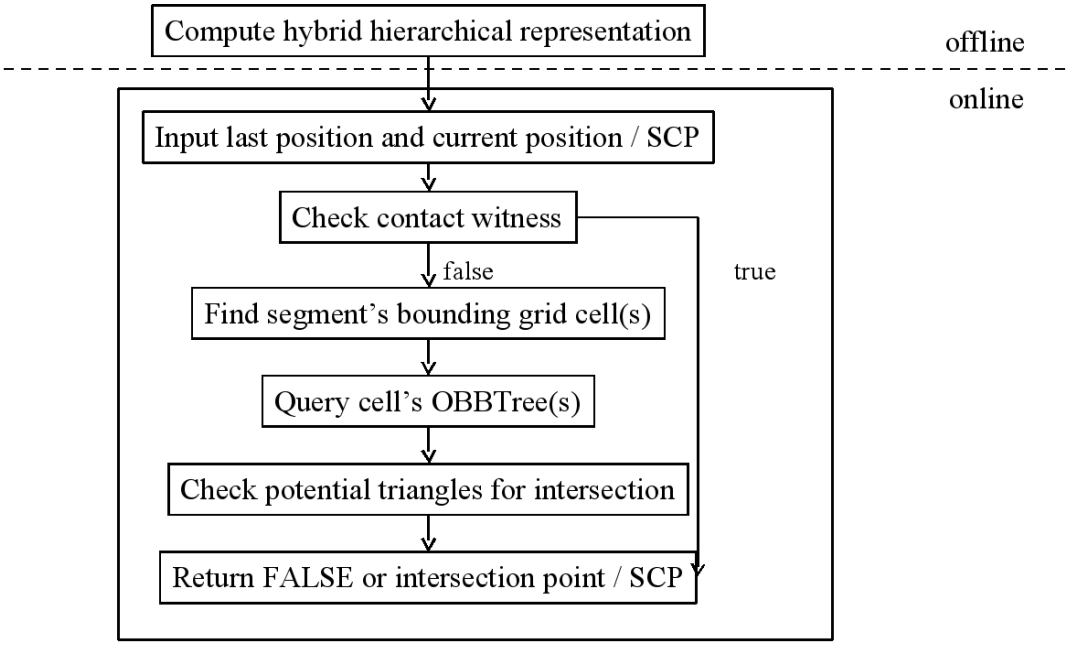


Figure 146: H-Collide system architecture

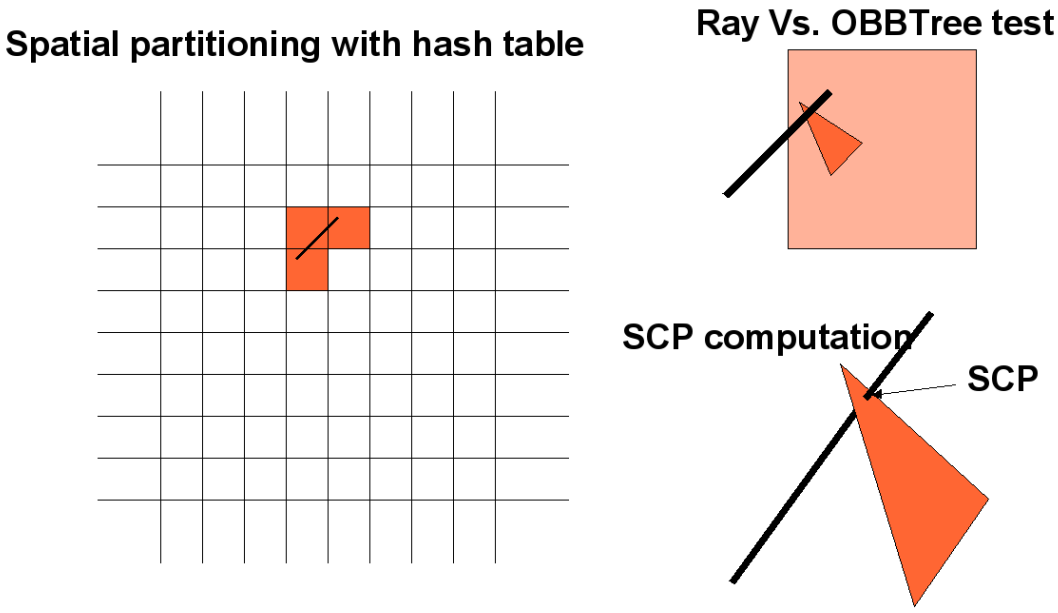


Figure 147: H-Collide overview

33.1 Introduction to Minkowski Sum and Duality

The Minkowski Sum computes the convolution of two shapes P and Q as the sum $P+Q = p+q|p \in P, q \in Q$ and the Minkowski Difference is a special case of the Minkowski sum of two convex shapes. It produces one shape grown by the shape of the other: $P-Q = p-q|p \in P, q \in Q$. Q is negated and then $P+(-Q)$ is computed.

The Minkowski sum of two convex objects is computed from the overlay of their Gauss maps. A Gauss map maps a feature (V,E,F) shown in Figure 152 in 3D to a unit sphere, according to surface normal. F is a face on the shape which produces a point in the Gauss map. E is an edge on the shape which produces a great arc in the Gauss map, and V is a vertex on the shape which produces a region bounded by great arcs.

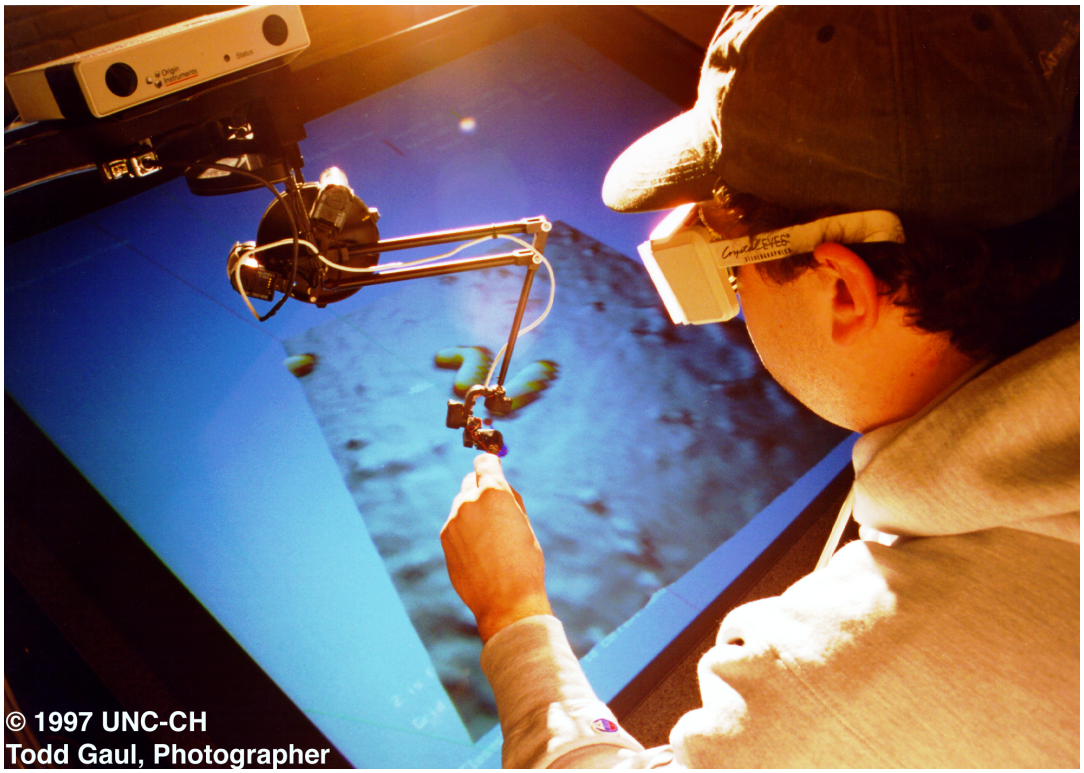


Figure 148: Exploring nano surfaces

33.1.1 Penetration Depth

The penetration depth between two intersecting polyhedra P and Q is defined as the minimum translational distance required for separating them. For intersecting polyhedra, the origin is contained in the Minkowski sum of P and $-Q$, 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]. this is summarized in Figure 153.

Figure 154 provides an overview of how to compute the Minkowski Difference from a Gauss Map.

Figure 155 illustrates incremental search of penetration depth. At a certain vertex in the overlay, the algorithm checks its corresponding penetration depth with that of the nearest neighbors. It then marches toward that vertex that minimizes the penetration depth. The actual Minkowski difference is locally computed when needed.

Possible Problems with the method are:

- Local minima: distance function from the origin to the surface of Minkowski difference can have multiple local minima. Good initialization provides a good result.
- Degeneracies:
  - Coplanar faces. Mapped to the same point in Gauss map. Treat them as a single point, and join the neighbors.
  - Central projection of Gauss map. Solved by local computation at each iteration.

Many researchers have restricted the computation of penetration depth to convex polyhedra. In computational geometry, Dobkin et al. [DHKS93] presented an algorithm for computing directional penetration depth, while Agarwal et al. [AGHP*00] introduced a randomized algorithm for computing the penetration depth between convex polyhedra. Cameron [Cam97] extended the GJK algorithm [GJK88] to compute bounds of the penetration depth, and van den Bergen [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 discretization. Fisher and Lin [FL01] estimate penetration depth using distance fields computed with fast marching level-sets. Hoff et al. [HZLM01] presented an image-based algorithm implemented on graphics hardware. On the other hand, Kim et al. [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.

(a) Painting of a tree

(b) Painting of a flowers

(c) Painting of a landscape

(d) Painting of cherry blossom

(e) Painting of a seascape

(f) A portrait

Figure 149: Examples of paintings generated with haptic brushes

Figure 150: 3D painting of a butterfly

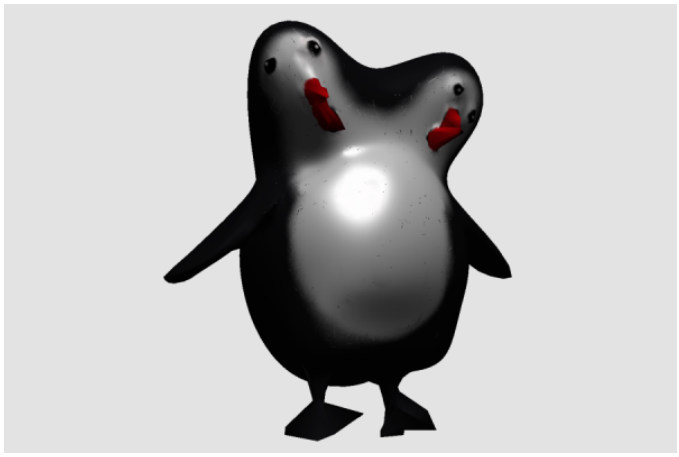


Figure 151: Multi-resolution haptic editing

Figure 152: Gauss Map

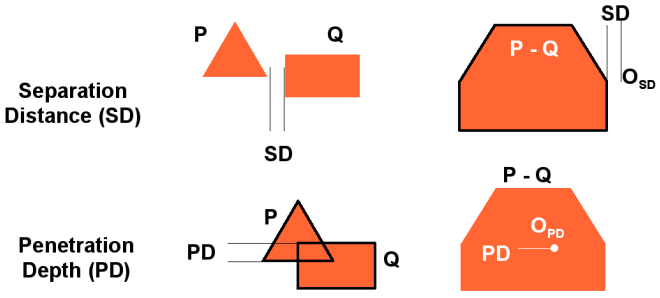


Figure 153: Penetration depth

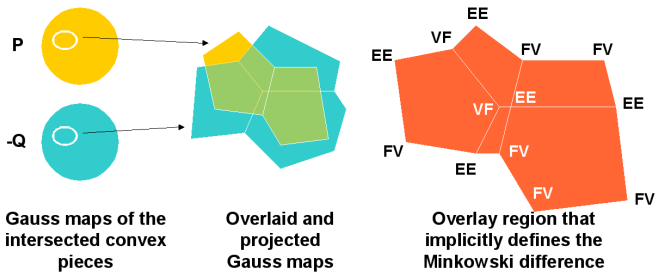


Figure 154: Minkowski Difference from Gauss Map

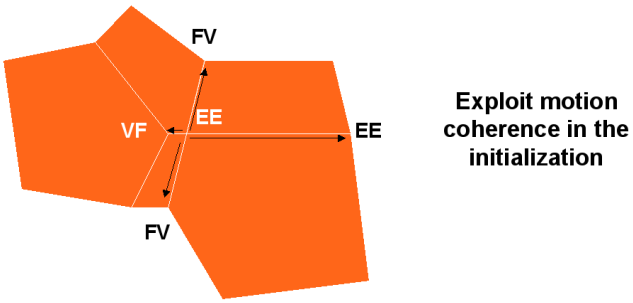


Figure 155: Incremental Search of penetration depth

33.2 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 initialization 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 realize 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 minimize 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 initialization features are given. In situations with high motion coherence, the penetration features from the previous frame are usually good initialization 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 realize the penetration depth. The combination of SWIFT++ and DEEP has also been tested for performing collision detection for haptic rendering [KOLM03].

33.3 Integration with SWIFT++ (a 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 $O(1)$ query time for each pair of convex hulls in most situations.

Acknowledgments

We would like to gratefully acknowledge the contributions of John Canny, Jonathan Cohen, Stephen Ehmann, Stefan Gottschalk, Young Kim, Eric Larsen, Dinesh Manocha, Brian Mirtich, Miguel Otaduy, Stephane Redon, Krish Ponamgi, and Nikunj Raghuvanshi.

Part X

High-Fidelity Haptic Interaction

As discussed earlier in the document, haptic rendering requires higher update rates than visual rendering, thereby limiting the complexity of the models that can be displayed. The fast growth of the geometric detail of the models used in computer graphics seems to even enlarge the gap between those models that can be graphically displayed in an interactive manner, from those that can be haptically displayed. Collision detection is precisely the main bottleneck for achieving effective haptic rendering of complex environments. However, psychophysics studies indicate that some dynamic and static factors which typically increase the cost of collision detection, such as high impact velocities or large contact areas, strongly limit our haptic ability to perceive the influence of high-resolution geometric details. These perceptual limitations call for the use of approximate collision detection algorithms that adaptively select the appropriate geometric resolution for contact computations. Hence, in the design of contact determination algorithms for high-fidelity haptic rendering, it is crucial to understand the psychophysics of touch and to account for perceptual factors.

Classic approximate techniques from computer graphics, such as levels of detail and texture mapping, can be adapted to the problem of collision detection. *Contact levels of detail* constitute a data structure that integrates in one dual hierarchy level-of-detail surface representations and bounding volume hierarchies for accelerating collision detection. A sensation-preserving simplification of geometric detail, along with error metrics for haptic rendering, enable the selection of the appropriate level of detail at each potential contact location independently. *Haptic textures* decouple the description of objects with high-resolution features into coarse parameterized meshes and texture images that store the fine geometric detail. Using haptic textures, collision detection can be formulated as a two-step problem that first identifies contact locations using coarse meshes, and then refines contact information accounting for the effect of geometric details. Along with perceptually-driven force models, haptic textures enable haptic rendering of the interaction between two complex textured models.

This part of the notes discusses relevant aspects in psychophysics research related to haptic perception, and describes in detail contact levels of detail and haptic textures, two perceptually-driven approximate collision detection algorithms especially designed for high-fidelity haptic rendering.

34 Psychophysics of Touch

The structure and behavior of human touch have been studied extensively in the field of psychology. The topics analyzed by researchers include characterization of sensory phenomena as well as cognitive and memory processes.

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

Knowing the exploratory procedure(s) associated with a particular object or surface property, researchers have studied the influence of various parameters on the accuracy and magnitude of sensory outputs. Perceptual studies on tactile feature detection and identification, as well as studies on texture or roughness perception are of particular interest for haptic rendering. In this section we summarize 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.

34.1 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] analyzed feature detection in robotic exploration, which can be regarded as a case of object-object interaction. They characterized geometric surface features based on the ratios of their curvatures to the radii of the robotic fingertips acquiring the surface data. They observed that a larger fingertip, which provides a larger contact area, can miss small geometric features. To summarize, the studies by Klatzky and Lederman [KL95] and Okamura and Cutkosky [OC99, OC01] lead to the 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 §35.

34.2 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 toward the characterization of cutaneous perception of textures. Katz [Kat89] suggested that roughness is perceived through a combination of spatial and vibratory codes during direct interaction with the skin. More recent evidence demonstrates that static pressure distribution plays a dominant role in perception of coarse textures (features larger than 1mm) [Led74, CJ92], but motion-induced vibration is necessary for perceiving fine textures [LS91, HR00]. As pointed out by Klatzky and Lederman [KL02], in object-object interaction roughness is encoded in vibratory motion transmitted to the subject.

In the last few years, Klatzky and Lederman have directed experiments that analyze the influence of several factors on roughness perception through a rigid probe. Klatzky et al. [KLH*03] distinguished three types of factors that may affect the perceived magnitude of roughness: 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 §36.

The experiments conducted by Klatzky and Lederman to characterize roughness perception [KL02] used a common setup: subjects explored a textured plate with a probe with a spherical tip, and then they reported a subjective measure of roughness. Plates of jittered raised dots were used, and the mean frequency of dot distribution was one of the variables in the experiments. The resulting data was analyzed by plotting subjective roughness values vs. dot 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 characterized by the maximum value of roughness and the value of texture spacing at which this maximum takes place.

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

34.3 Combination of Visual and Haptic Display

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

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

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

The design of contact determination algorithms can also benefit from existing studies on the visual perception of collisions in computer animations. O'Sullivan and her colleagues [ORC99, OD01, ODGK03] have investigated different factors affecting visual collision perception,

including eccentricity, separation, distractors, causality, and accuracy of simulation results. Basing their work on a model of human visual perception validated by psychophysical experiments, they demonstrated the feasibility of using these factors for scheduling interruptible collision detection among large numbers of visually homogeneous objects.

35 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 156. Findings from tactual perception and spatial recognition [KL95, OC99, OC01] demonstrate that large contact areas reduce the perceptibility of fine surface features.

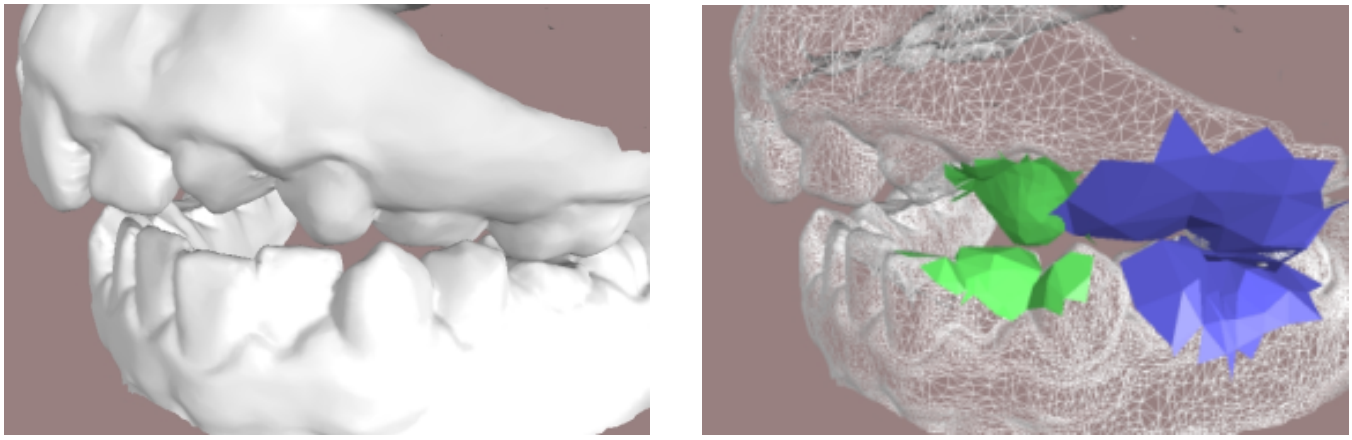


Figure 156: **Multi-resolution Collision Detection Using CLODs.** *Left: Moving jaws in contact, rendered at their highest resolution. Right: The appropriate object resolution (shown in blue and green) is adaptively selected at each contact location, while the finest resolution is displayed in wireframe.*

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 organized as follows. §35.1 presents the motivation and design goals of a multi-resolution collision detection algorithm for haptic rendering. §35.2 introduces the data structure of CLODs, and §35.3 presents a particular implementation based on convex hulls, followed by the description of sensation preserving simplification. §35.4 explains how contact levels of detail are used in runtime collision detection. §35.5 presents the experiments and results and, last, §35.6 concludes with a discussion of limitations.

35.1 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.

35.1.1 Haptic Perception of Surface Detail

§34.1 summarized 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 §35.3) of a polygonal model. That is, higher resolution around a local area implies that the width and length of the geometric surface features in that neighborhood are smaller, and vice versa. The concept of "height" is extended to describe the amount of surface deviation between polygonal representations of a model at different resolutions.

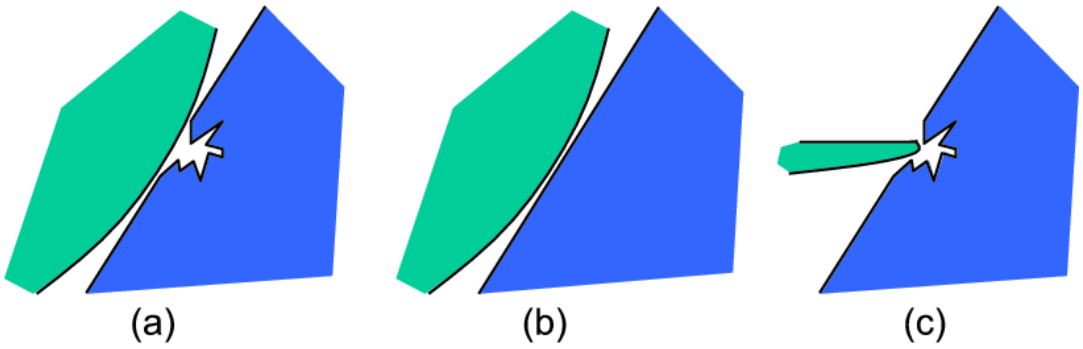


Figure 157: **Contact area and resolution:** *(a) high-resolution model with large contact area; (b) low-resolution model with large contact area; (c) high-resolution model with small contact area.*

Figure 157 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 §35.3, we show how haptic error metrics drive the offline construction of CLODs, and in §35.4, we show how they are used in runtime contact queries.

35.1.2 Hierarchical Collision Detection

The running time of any collision detection algorithm depends on both the input and output sizes of the problem. Given two polyhedra, characterized by their combinatorial complexity of m and n polygons, the collision detection problem can have an output size as large as $O(mn)$. Please refer to §28 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 §28.2, 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 initializing the collision query in the next time step. The overall cost of a collision test is proportional to the number of nodes in the front of the BVTT.

When large areas of the two objects are in close proximity, a larger portion of the BVTT front is close to the leaves, and it consists of a larger number of nodes. The size of the front also depends on the resolutions with which the objects are modeled; higher resolutions imply a deeper BVTT. To summarize, the cost of a collision query depends on two key factors: the size of the contact area and the resolutions of the models. As observed in §35.1.1, 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.

35.1.3 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.

§35.2 describes the generic data structure employed in CLODs. The implementation of CLODs, however, requires the selection of one particular type of bounding volume. §35.3 presents the construction of CLODs using convex hulls as the bounding volumes.

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

1. **Minimize 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).

35.2 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.

35.2.1 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.

35.2.2 Description of the Data Structure

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

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

The tree T of BVs, together with the Hausdorff distances, serves as the BVH for culling purposes in collision detection. Directed Hausdorff distances are necessary because, in the definition of CLODs, the set of BVs associated with one particular LOD may not bound the surface of previous LODs. Hausdorff distances are used to perform conservative collision tests, as will be later explained in §35.4.2.

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.

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**

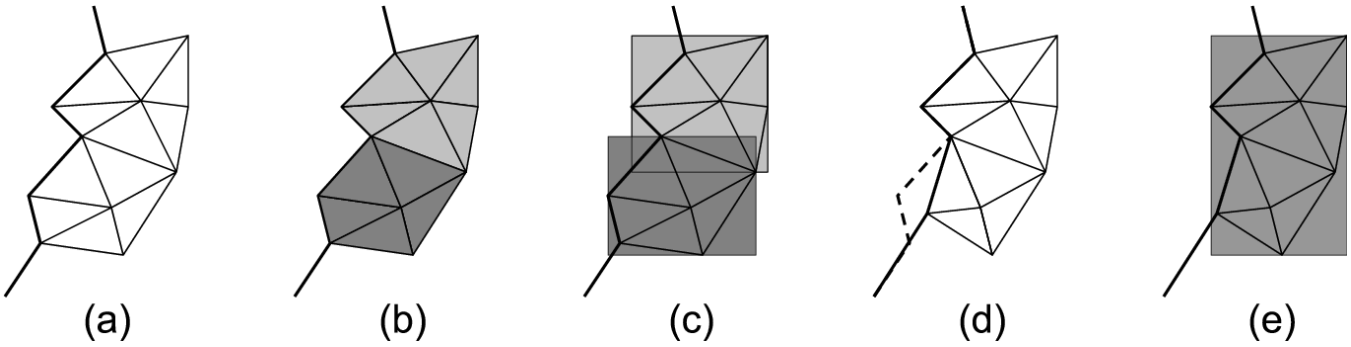


Figure 158: **Construction of generic CLODs:** *(a) Initial surface; (b) Clusters of triangles; (c) BVs for each cluster; (d) Mesh simplification; (e) BV of the union of clusters after some conditions are met.*

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.

35.2.3 Generic Construction of Contact Levels of Detail

The process of creating the CLODs, depicted in Figure 158, starts by grouping the triangles of the original surface into clusters. The sizes and properties of these clusters depend on the type of BV that is used for the BVH, and will be such that the performance of the collision query between two BVs is optimized. The next step in the creation of CLODs is to compute the BV of each cluster. This initialization is followed by a mesh decimation process along with bottom-up construction of the BVH, carried out by merging clusters and computing the BV of their union.

The atomic simplification operations need to satisfy the following conditions:

- **Constraints imposed by the BVH:** The containment of surface geometry inside the BVs has to be preserved after each simplification operation. This condition may impose topological and/or geometric constraints.
- **Design requirements to achieve better efficiency:** Clusters can be joined when certain conditions are met. The BVH will be more effective in collision pruning if these conditions are taken into account when designing the atomic simplification operations.

In §35.3, 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.

35.3 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.

35.3.1 Selection of Convex Hulls as Bounding Volumes

Overlap tests between convex hulls can be executed in expected constant time with motion coherence [LC91, Mir98, 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 maximize 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.

35.3.2 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.

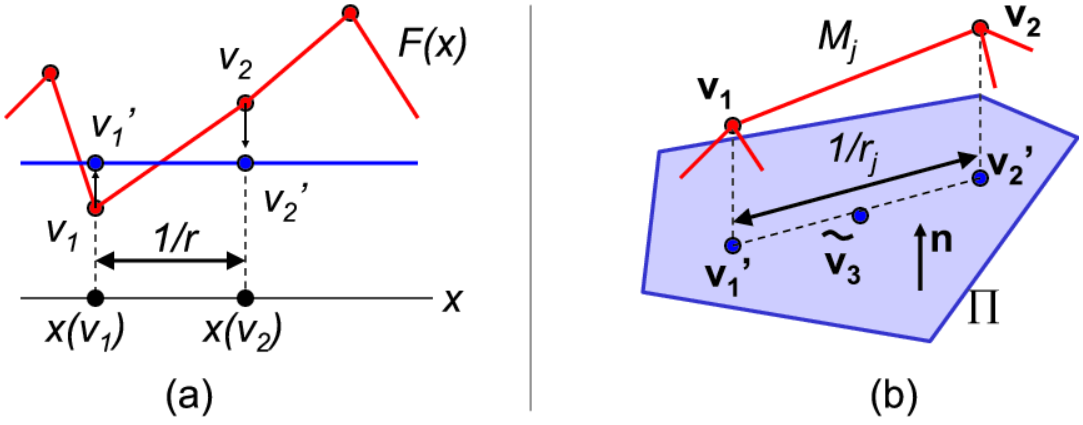


Figure 159: **Definition of Resolution.** *(a) Resolution r defined in the 1D setting; (b) Resolution for irregular meshes.*

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 159-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 159-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.

$$r = \frac{1}{\|(\mathbf{v}_1 - \mathbf{v}_2) - ((\mathbf{v}_1 - \mathbf{v}_2) \cdot \mathbf{n}) \cdot \mathbf{n}\|}. \tag{43}$$

35.3.3 Construction of Contact Levels of Detail

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

The atomic simplification operations must take into account the convexity constraints. After each operation, the union of every pair of neighboring convex patches is tested for convexity. If the union is a valid convex patch itself, the involved patches are merged and the convex hull of the union is computed. All the BVs in LOD M_j that are merged to a common BV $C_{j+1} \in M_{j+1}$ during sensation preserving simplification will have C_{j+1} as their parent in the BVH. 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 practice, due to topological and geometric constraints that limit the amount of simplification, and which cannot be removed by local operations. In such cases, the hierarchy is completed by unconstrained pairwise merging of patches [EL01]. The levels of the hierarchy created in this manner, denoted as "free" LODs, cannot be used to report contact information in multi-resolution collision queries, but are necessary to complete the BVH.

The following algorithm gives the pseudo code for the process of sensation preserving simplification:

```
Compute surface convex decomposition

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

Figure 160: **Pseudo Code of the Sensation Preserving Simplification Loop**

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 prioritizing edges and selecting the position of the resulting vertices: minimization of energy functions [Hop96], optimization approaches [LT98], quadric error metrics for measuring surface deviation [GH97a], and more. None of these approaches, however, meets the convexity constraints or takes into account the factors that maximize the efficiency of CLODs. Another possibility is to employ the *progressive hulls* representation [SGG*00], which maintains the interesting property of containment for collision detection. In the progressive hulls representation, however, successive LODs are not simply 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 §35.2.2, 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 prioritized based on their resolution.

In the construction of CLODs, and as part of the initialization, 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) 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 maximize the resolution at which LODs are generated. As explained in §35.4, 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 maximizing the resolution at which LODs are generated. This is the basis for selecting edge resolution as the priority for edge collapses.

35.3.4 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 initialization 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 minimization of second order divided differences [GSS99].

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

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

The unconstrained relaxation step resembles, intuitively, the minimization of dihedral angles, without much affecting the shape of the triangles [GSS99]. 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 minimization 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 neighborhood of $\mathbf{v}_3$ may change from convex to reflex and vice versa. Interior edges of convex patches are convex before the filtered edge collapse and must remain convex after it. These constraints can be expressed as linear constraints in the position of $\mathbf{v}_3$.

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 161. 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 161, the convexity

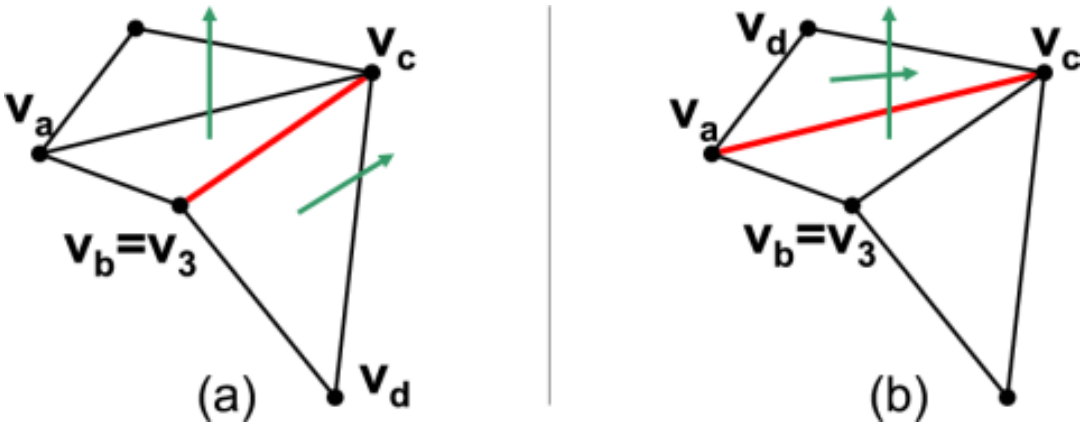


Figure 161: **Local Convexity Constraints.** *Assignment of vertices* $\mathbf{v}_a$, $\mathbf{v}_b$, $\mathbf{v}_c$ *and* $\mathbf{v}_d$ *for an interior edge (a) incident on* $\mathbf{v}_3$, *and (b) opposite to* $\mathbf{v}_3$.

constraint of an edge can be expressed as a negative volume for the parallelepiped defined by the adjacent triangles:

$$((\mathbf{v}_b - \mathbf{v}_a) \times (\mathbf{v}_c - \mathbf{v}_a)) \cdot (\mathbf{v}_d - \mathbf{v}_a) \leq 0. \tag{44}$$

The convexity constraints can be satisfied by formulating an optimization program in which $\mathbf{v}_3$ is constrained to the segment between the position that minimizes 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:

$$\begin{aligned} \mathbf{v}_3 &= u \cdot \hat{\mathbf{v}}_3 + (1-u) \cdot \tilde{\mathbf{v}}_3, \\ u &\geq 0, \\ u &\leq 1. \end{aligned} \tag{45}$$

The convexity constraints in Eq. 44 can be rewritten based on constants A and B as:

$$\begin{aligned} A \cdot u + B \geq 0, \quad \text{where} \\ A = ((\mathbf{v}_d - \mathbf{v}_a) \times (\mathbf{v}_c - \mathbf{v}_a)) \cdot (\hat{\mathbf{v}}_3 - \tilde{\mathbf{v}}_3), \\ B = ((\mathbf{v}_d - \mathbf{v}_a) \times (\mathbf{v}_c - \mathbf{v}_a)) \cdot (\tilde{\mathbf{v}}_3 - \mathbf{v}_a). \end{aligned} \tag{46}$$

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 optimization 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 162. The resulting vertex $\mathbf{v}_3$ is reassigned to this position.

35.3.5 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 §35.4, 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:

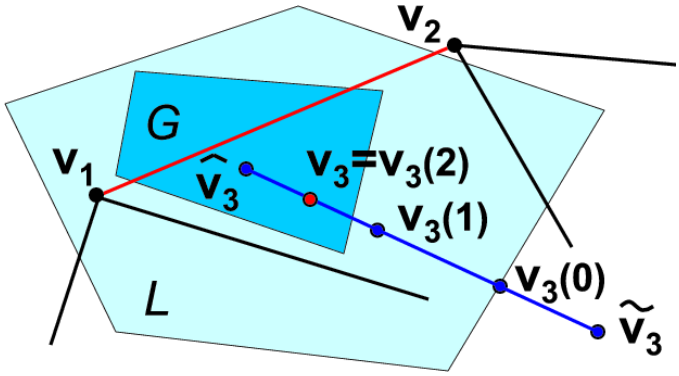


Figure 162: **Filtered Edge Collapse with Convexity Constraints.** *The figure shows a filtered edge collapse in which bisection search is required to find a position that meets the convexity constraints. G and L represent feasible regions of global and local constraints respectively.*

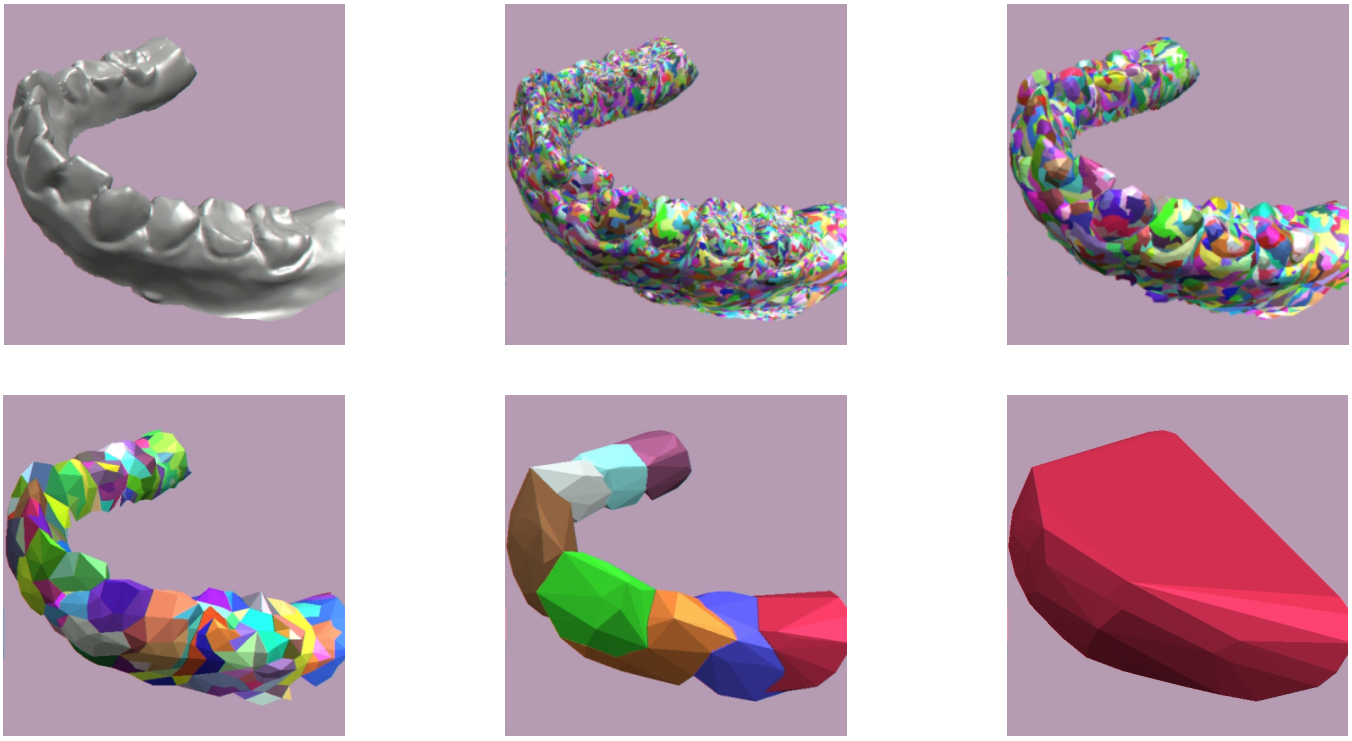


Figure 163: **CLODs of a Lower Jaw.** *From left to right and top to bottom, original mesh, M_0, and convex patches of M_0, M_3, M_6, M_{11}, and M_{14}.*

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 **v** of the initial mesh M_0, the support area *D* is computed as the projected area onto the tangent plane of **v** of the faces incident to **v**, such that they are within a certain distance tolerance from **v** along the direction of the normal **n** of **v** and their normal lies inside a normal cone of **n**. For this computation, one may use the same distance tolerance as the one used for contact queries (see §35.4). 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*.

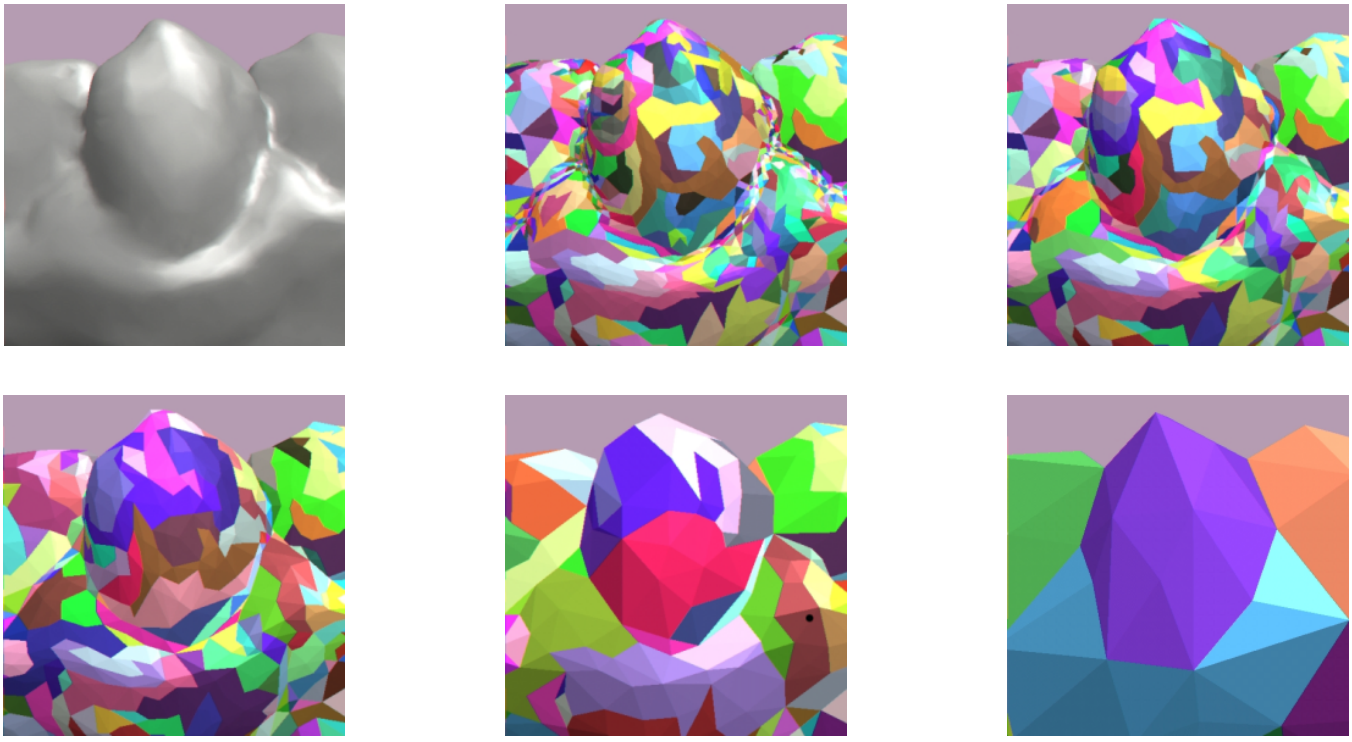


Figure 164: **Detail View of the CLODs of a Lower Jaw.** *From left to right and top to bottom, original mesh, M_0, and convex patches of M_0, M_1, M_2, M_4, and M_7.*

35.3.6 Examples of Contact Levels of Detail

Figure 163 shows several of the LODs obtained when processing a model of a lower jaw (see §35.5 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 necessarily bound the original surface, as may be deduced from the figures. As described later in §35.4.2, 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 164 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.

35.4 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.

35.4.1 Contact Query between Two Objects

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

For haptic rendering purposes, 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, Mir98, 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 localized penetration depth values [KOLM03].

As mentioned in §35.1.2, 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.

35.4.2 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 156. 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 §35.4.3, 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:

$$d_{ab} = d + h(a) + h(b), \tag{47}$$

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 §35.3.5, these Hausdorff distances can be precomputed during the process of sensation preserving simplification.

Analysis of the Bounding Volume Test Tree

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

Using CLODs, when the distance query and the selective refinement test return true for a node ab of the BVTT, 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 165. Nodes of the BVTT at coarser resolution are closer to the root. A resolution-based ordering of the BVTT is a key factor for maximizing the performance of runtime collision detection, because CLODs with lower resolution and larger error are stored closer to the root of the BVTT. Descending along the BVTT has the effect of refining the CLODs. The resolution-based ordering of CLODs of two different objects is possible because the definition of resolution presented in §35.3.2 is an object-independent absolute metric. If objects are scaled, the value of resolution must be scaled accordingly.

As pointed out in §35.3.3, 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 166 by the line λ.

As indicated in §35.1.2, 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 166. 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.

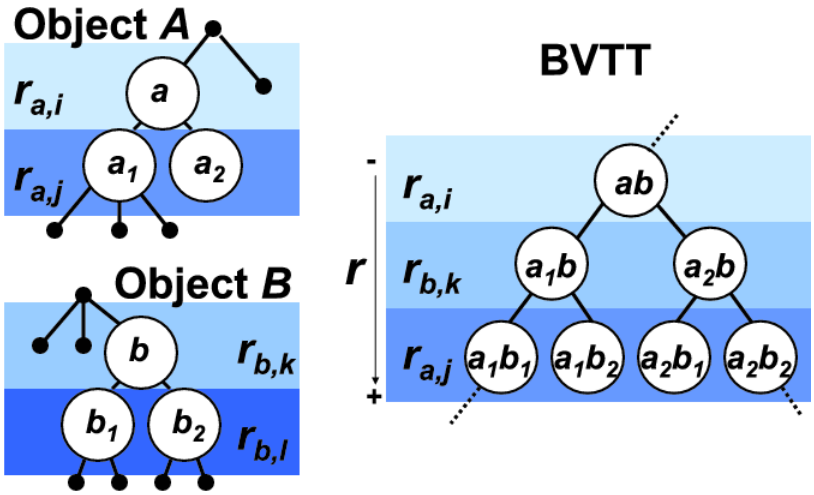


Figure 165: **Resolution-Based Ordering of the Bounding Volume Test Tree.** *A node splitting policy based on CLOD resolution implies a BVTT in which levels are sorted according to increasing resolution. Descending on the BVTT has the effect of increasing CLOD resolution.*

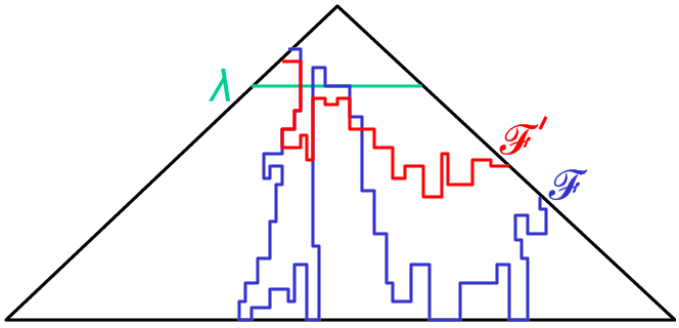


Figure 166: **Generalized Front Tracking of the BVTT.** *The front of the BVTT for an exact contact query, $\mathbb{F}$, is raised up to the new front $\mathbb{F}'$ using CLODs, since the recursive distance queries can stop at coarser resolutions. λ indicates the free CLODs, constructed by unconstrained merging of convex patches.*

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 166. This approach results in a much faster processing of contact queries and, ultimately, it enables 6-DoF haptic rendering of complex objects.

35.4.3 Selective Refinement and Error Metrics

As discussed in §35.1.1, 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:

$$\phi_a = \frac{s_a}{r_a^2}, \tag{48}$$

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:

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

$$D = \max(D_a, D_b). \tag{49}$$

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 §35.3.5, 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 §35.5. Based on experiments described in §35.5.2, 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,

$$s_0 = \frac{N \cdot l \cdot f}{n \cdot i}. \tag{50}$$

Constant Frame Rate

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

35.4.4 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 §35.5.

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.

35.5 Experiments and Results

In this section we describe experiments conducted to test and analyze 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.

35.5.1 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 normalized resolution (for unit object radius) of the finest and coarsest CLODs, and the number of "free" and total CLODs. As described in §35.3.3, 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 168, 169, and 170.

35.5.2 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.

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 167. 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.

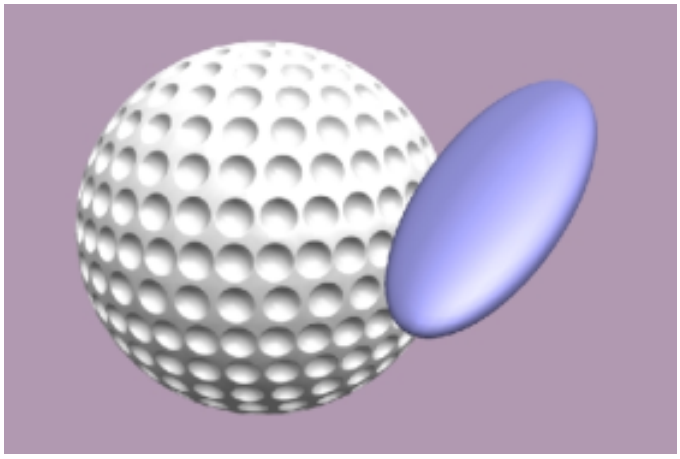


Figure 167: **Exploration of a Multi-resolution Golf Ball with an Ellipsoid.** *Scenario of the experiments for identifying haptic error tolerances with CLODs.*

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.

35.5.3 Performance Experiments in 6-DoF Haptic Rendering

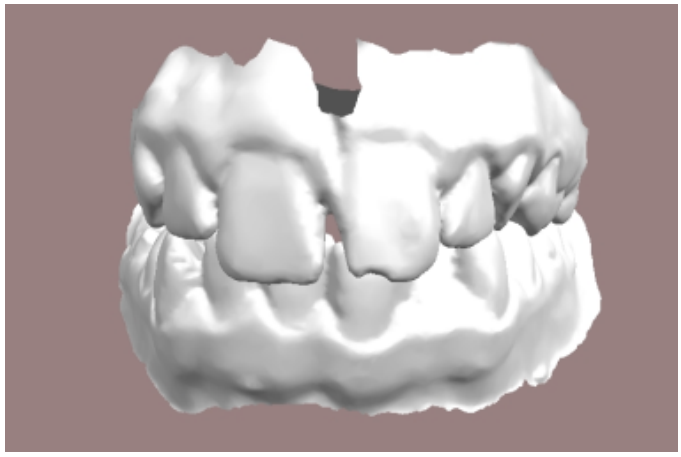


Figure 168: **Upper and Lower Jaws.** *A benchmark scenario for 6-DoF haptic rendering using CLODs.*

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

- Moving upper and lower jaws (See Figure 168).
- Interlocking ball joints (See Figure 169).
- Golf club tapping a golf ball (See Figure 170).

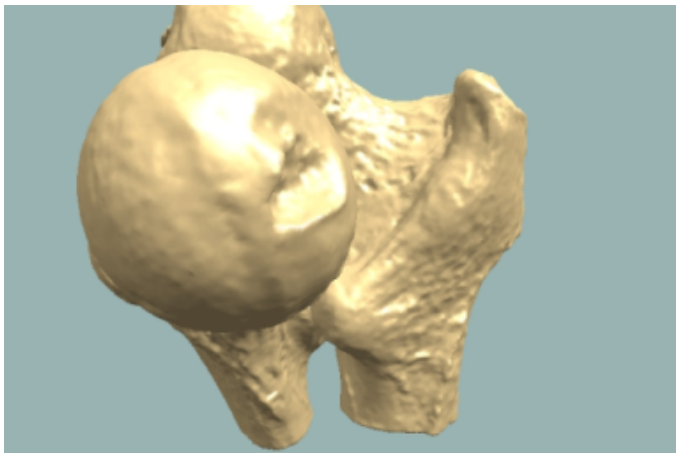


Figure 169: **Ball Joints.** *A benchmark scenario for 6-DoF haptic rendering using CLODs.*

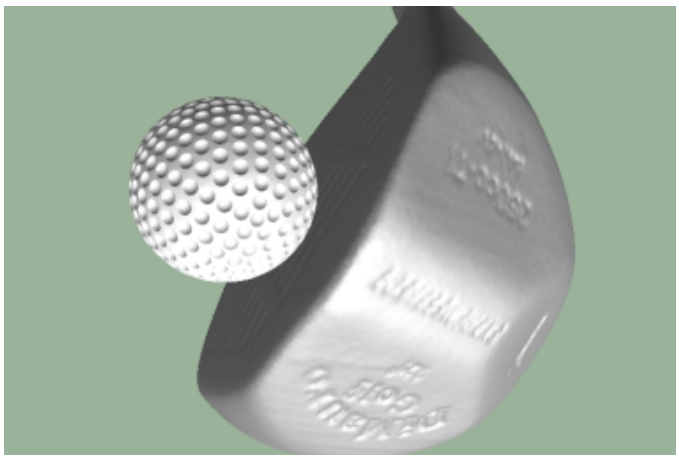


Figure 170: **Golf Club and Ball.** *A benchmark scenario for 6-DoF haptic rendering using CLODs.*

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

Contact forces and running time are analyzed 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 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

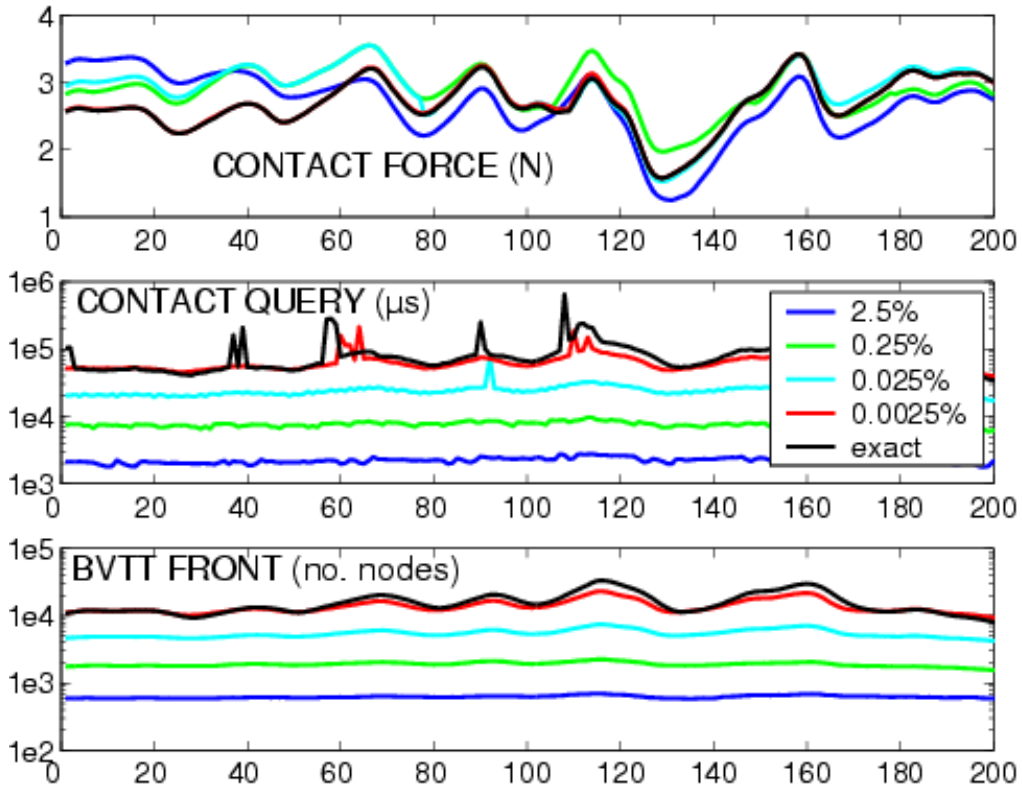


Figure 171: **Contact Profile for Moving Jaws.** *Top: The profiles of the contact forces displayed using CLODs, with varying error tolerances up to 2.5% of the radii of the jaws, all show very similar patterns. This similarity implies that the sensations of shape provided to the user are nearly identical.*
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.

Figure 171 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 172 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 171 and Figure 172 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.
- 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.

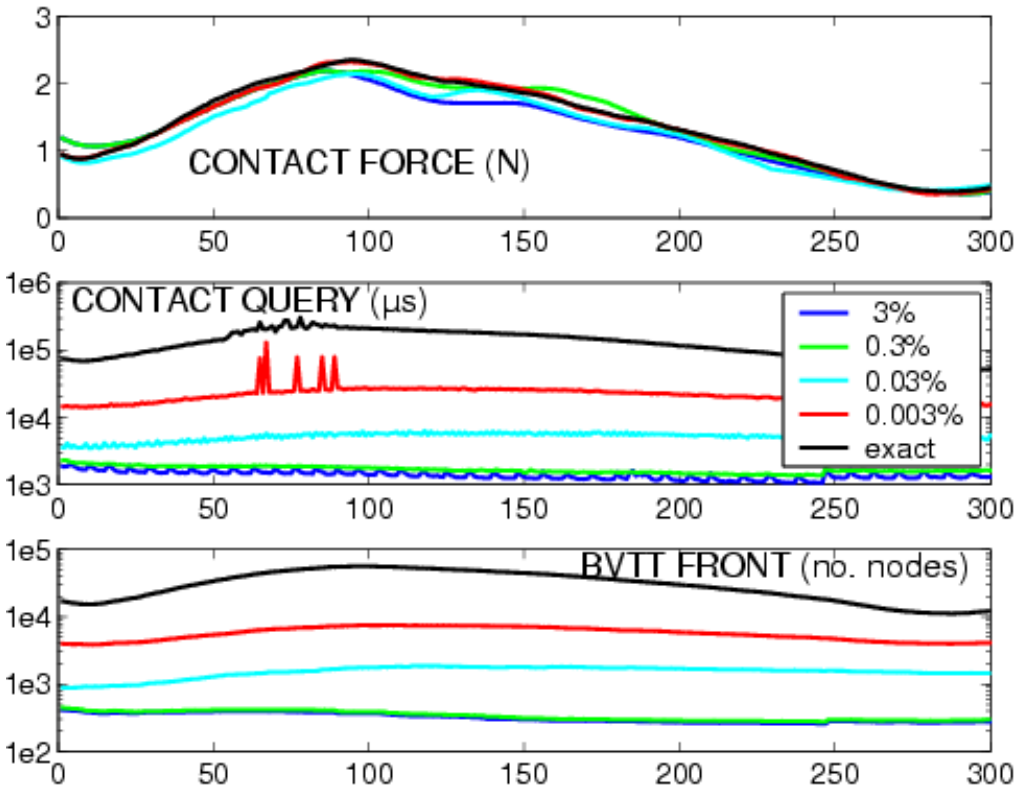


Figure 172: **Contact Profile for Golf Scene.** *Top: The profiles of the contact forces displayed using CLODs, with varying error tolerances up to 3% of the radius of the ball, show nearly identical patterns.*
Middle: A log plot of contact query time using CLODs with various error tolerances shows more 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.

35.6 Discussion and Limitations

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

35.6.1 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.

35.6.2 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.

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 minimize surface deviation. The elimination of detail benefits the creation of the BVH and does not detract from the output quality of haptic rendering, since the filtered detail is quantified and taken into account in runtime collision detection.

Multiresolution representations obtained through mesh decimation techniques are not able by themselves to support efficient contact queries.

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

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

- **Lack of containment.** As introduced in §35.2.2, 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 §35.4.2, 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 §35.3.3, 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 §35.3.3, 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 maximized if the topological and geometric constraints can be removed during the creation of the CLODs.
- **Static LODs.** A surface patch may undergo several atomic simplification operations between two consecutive CLODs, which introduce discontinuities in the multi-resolution representation. In §35.4.4, 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 §35.3.1).

35.6.3 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.

36 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 §35, we have described *contact levels of detail* (CLODs) [OL03b, OL03a], a multi-resolution collision detection algorithm especially designed for 6-DoF haptic rendering that minimizes the computational impact of collision detection and selects the appropriate object resolution at each contact location. CLODs, however, filter out high-resolution geometric features, thus ignoring texture effects arising in sliding, rolling, and twisting motion. This 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] synthesized 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.

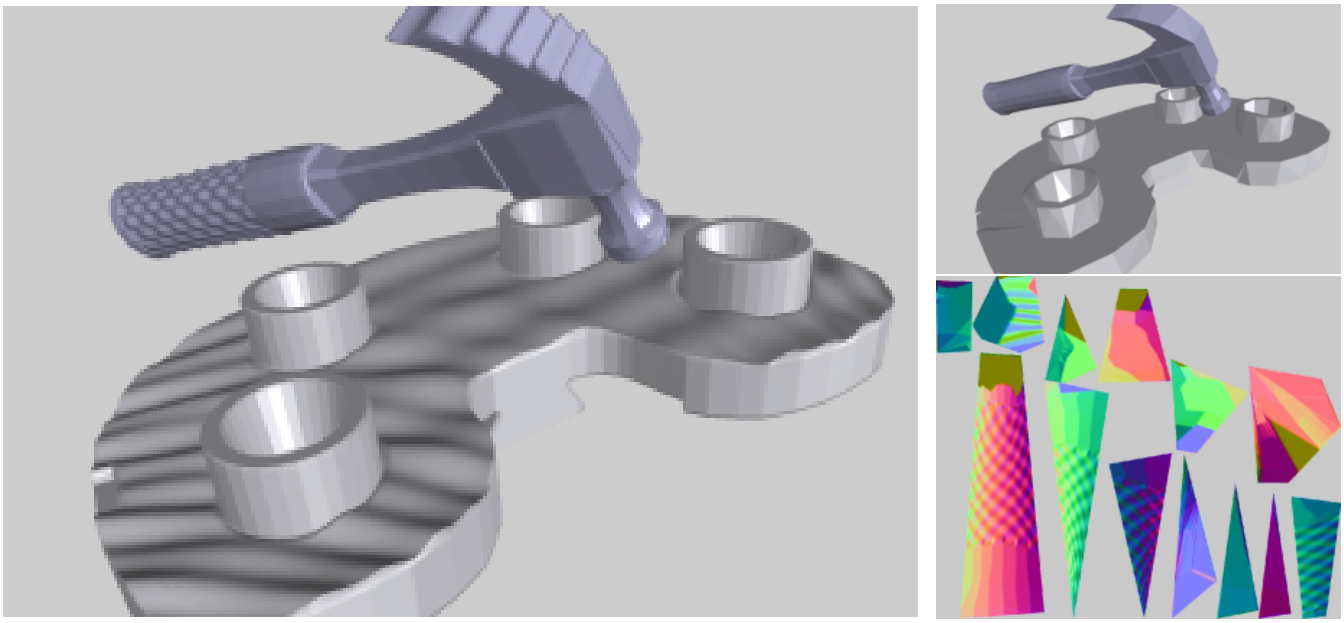


Figure 173: **Haptic Rendering of Interaction between Textured Models.** *Top: high-resolution textured hammer (433K polygons) and CAD part (658K polygons); Bottom left: low-resolution models (518 & 720 polygons); Bottom right: hammer texture with fine geometric detail.*

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 173. 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 analyzed 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 organized as follows. §36.1 defines the notation used throughout the section and key terminology related to the concept of penetration depth. §36.2 presents the foundations of the rendering algorithm and the force model, which is described in §36.3. §36.4 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 §36.5, and §36.6 concludes with a discussion on limitations of this work.

36.1 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.

36.1.1 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 $g : D \to S, D \subset \mathbb{R}^2$, as $g(p_u,p_v) = \mathbf{p}_{xyz}$, where:

$$\mathbf{p}_{xyz} = g(p_u,p_v) = (\mathbf{u} \ \mathbf{v} \ \mathbf{n})(p_u \ \ p_v \ \ h(p_u,p_v))^T . \tag{51}$$

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:

$$h(p_u,p_v) = \mathbf{n} \cdot g(p_u,p_v). \tag{52}$$

36.1.2 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 174-b). 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 174-c). The penetration depth between two intersecting surface patches will be referred to as *local penetration depth*.

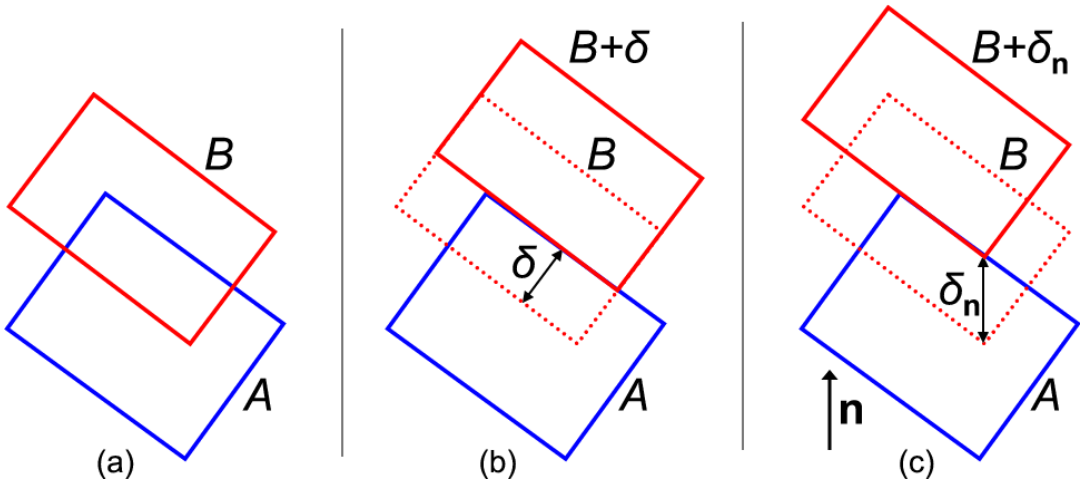


Figure 174: **Definitions of Penetration Depth.** *(a) Intersecting objects A and B, (b) global penetration depth* δ*, and (c) directional penetration depth* δ_n *along* $\mathbf{n}$.

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 parameterized by orthographic projection along $\mathbf{n}$, as expressed in §36.1.1. The parameterization yields mappings $g_A : D_A \to S_A$ and $g_B : D_B \to S_B$, as well as height functions $h_A : D_A \to \mathbb{R}$ and $h_B : D_B \to \mathbb{R}$. 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 175 by a 2D example.

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

$$\delta_{\mathbf{n}} = \max_{(u,v)\in(D_A \cap D_B)} (h_A(u,v) - h_B(u,v)) . \tag{53}$$

36.2 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 §34.2, we have summarized 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.

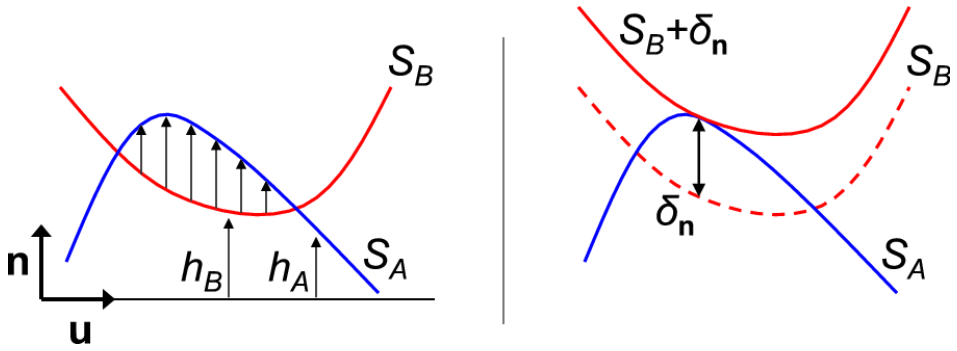


Figure 175: **Penetration Depth of Height Fields.** *Directional penetration depth of surface patches expressed as height difference.*

36.2.1 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 176. 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.

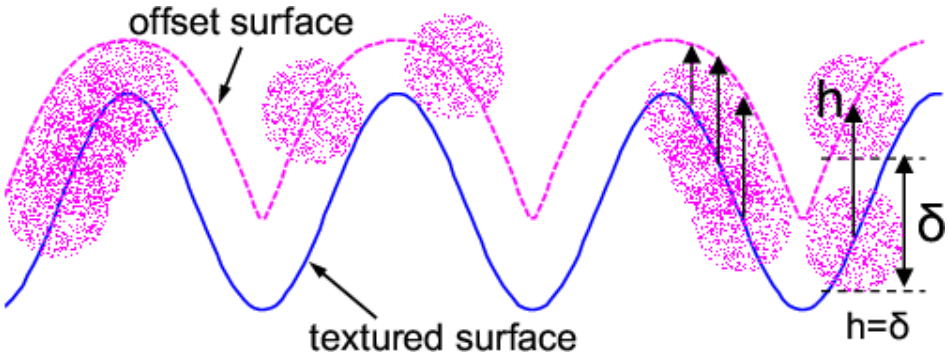


Figure 176: **Offset Surfaces.** *Left: offset surface computed as the convolution of a surface with a sphere; Centre: sphere whose trajectory traces an offset surface; Right: correspondence between vertical penetration depth (δ) and height of the offset surface (h).*

In the design of a force model for haptic texture rendering, one faces the question: How can the concept of offset surface be generalized 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 176. In this situation, the probe penetrates the textured surface. The *vertical penetration depth* δ is the vertical translation required to separate the probe from the textured surface, and it is the same as the height of the offset surface *h*. Unlike the height of offset surfaces, (directional) penetration depth is a metric that can be generalized to the interaction between arbitrary surfaces.

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

Since, for spherical probes, perception of roughness is tightly coupled with the undulation of the traced offset surface, 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 generalization of the concept used in 3-DoF haptic rendering.

36.2.2 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 Characterize each contact by a pair of contact points on the patches and a penetration direction **n**.
2. Refine this contact information using detailed geometric information stored in haptic textures.
   2.1 For each contact, compute approximate directional penetration depth along **n**, using haptic textures.
   2.2 Compute force and torque, using a novel force model for texture rendering.

The 6-DoF haptic texture rendering algorithm presented in this 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 parameterized and, in case of using CLODs, the parameterization must be consistent across all levels of the hierarchy, and distortion must be minimized. One possible approach is to integrate parameterization 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 §35.

36.3 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.

36.3.1 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 modeled explicitly. Nevertheless, Otaduy et al. [OJSL04] analyzed the dynamic behavior of the force model, observing that vibratory motion produced by simulated forces behaves in a way similar to physical roughness perception. The experiments are described in detail in §36.5.1.

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.

36.3.2 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:

$$U = \frac{1}{2} k \delta^2. \tag{54}$$

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

$$\begin{pmatrix} \mathbf{F} \\ \mathbf{T} \end{pmatrix} = -\nabla U = -k\delta\,(\nabla\delta)\,, \tag{55}$$

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 §36.2.2, 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. 55 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. 55 reduces to:

$$\begin{pmatrix} F_u & F_v & F_n & T_u & T_v & T_n \end{pmatrix}^T = -k\delta_{\mathbf{n}} \begin{pmatrix} \frac{\partial \delta_{\mathbf{n}}}{\partial u} & \frac{\partial \delta_{\mathbf{n}}}{\partial v} & 1 & \frac{\partial \delta_{\mathbf{n}}}{\partial \theta_u} & \frac{\partial \delta_{\mathbf{n}}}{\partial \theta_v} & \frac{\partial \delta_{\mathbf{n}}}{\partial \theta_n} \end{pmatrix}^T, \tag{56}$$

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:

$$\begin{aligned} \mathbf{F}_A &= (\mathbf{u}\ \mathbf{v}\ \mathbf{n})\,(F_u\ F_v\ F_n)^T, \\ \mathbf{T}_A &= (\mathbf{u}\ \mathbf{v}\ \mathbf{n})\,(T_u\ T_v\ T_n)^T. \end{aligned} \tag{57}$$

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

Generalizing Minsky's approach [Min95], the tangential forces F_u and F_v are proportional to the gradient of penetration depth. However, 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].

36.3.3 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:

$$\frac{\partial \delta_{\mathbf{n}}}{\partial u} = \frac{\delta_{\mathbf{n}}(u+\Delta u, v, n, \theta_u, \theta_v, \theta_n) - \delta_{\mathbf{n}}(u-\Delta u, v, n, \theta_u, \theta_v, \theta_n)}{2\Delta u}, \tag{58}$$

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.

36.4 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.

36.4.1 Directional Penetration Depth

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

As expressed in §36.2.2, 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 §36.1.1. Given a rotated reference system $\{\mathbf{u},\mathbf{v},\mathbf{n}\}$, S_A and $\hat{S}_A$ are projected orthographically along $\mathbf{n}$ onto the plane $(\mathbf{u},\mathbf{v})$. This projection yields mappings $g_A : D_A \rightarrow S_A$ and $\hat{g}_A : \hat{D}_A \rightarrow \hat{S}_A$. 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 $f_A \circ \hat{g}_A : \bar{D}_A \rightarrow S_A$ (See Figure 177). From Eq. 52, an approximate height function $\hat{h}_A : \bar{D}_A \rightarrow \mathbb{R}$ is defined as:

$$\hat{h}_A(u,v) = \mathbf{n} \cdot (f_A \circ \hat{g}_A(u,v)). \tag{59}$$

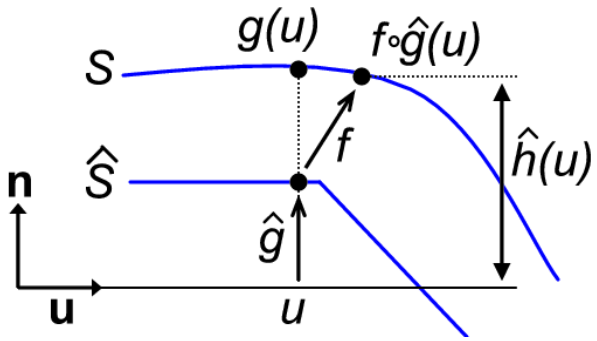


Figure 177: **Approximate Height Function.** *Height function of a surface patch approximated by a composite mapping function.*

Given approximate height functions $\hat{h}_A$ and $\hat{h}_B$, a domain $D = \bar{D}_A \cap \bar{D}_B$, and Eq. 53, the directional penetration depth $\delta_{\mathbf{n}}$ of S_A and S_B can be approximated by:

$$\hat{\delta_{\mathbf{n}}} = \max_{(u,v)\in D} \left(\hat{h}_A(u,v) - \hat{h}_B(u,v)\right). \tag{60}$$

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

36.4.2 Computation on Graphics Hardware

As shown in Eq. 56, 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. 58, 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. 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. 59 and 60 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 maximize performance. The input haptic texture f is stored as a floating point texture.

Max Search

The *max* function in Eq. 60 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.

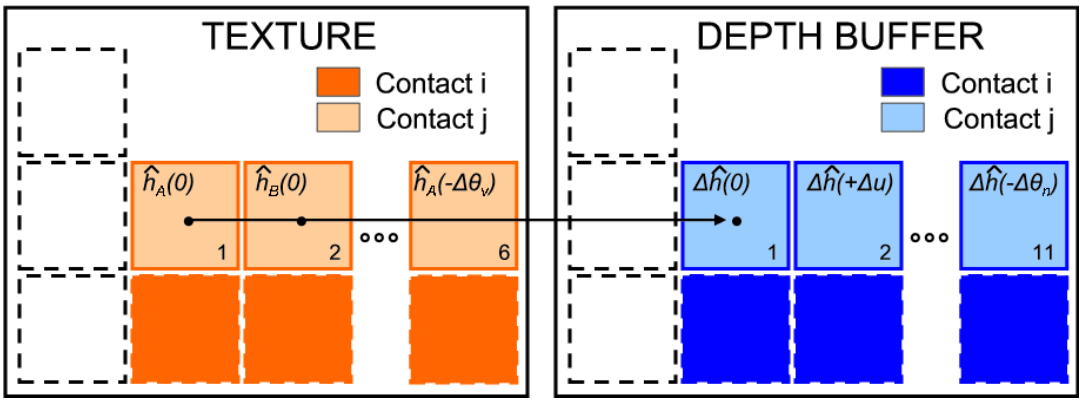


Figure 178: **Tiling in the GPU.** *Tiling of multiple height functions and contacts to minimize context switches between target buffers.*

Gradient Computation

The height functions $\hat{h}_A(\pm\Delta u)$, $\hat{h}_A(\pm\Delta v)$ and $\hat{h}_A(\pm\Delta\theta_n)$ may be obtained by simply translating or rotating $\hat{h}_A(0)$. As a result, only 6 height functions $\hat{h}_A(0)$, $\hat{h}_B(0)$, $\hat{h}_A(\pm\Delta\theta_u)$ and $\hat{h}_A(\pm\Delta\theta_v)$ need to be computed for each pair of contact patches. These 6 height functions are tiled in one single texture t to minimize context switches and increase performance (See Figure 178).

Moreover, the domain of each height function is split into 4 quarters, each of which is mapped to one of the RGBA channels. This optimization exploits vector computation capabilities of fragment processors. As shown in Figure 178, 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 optimized. 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 178. The cost of *max search* operations is further minimized by performing occlusion queries on all contacts in parallel.

36.5 Experiments and Results

Otaduy et al. [OL04, OJSL04] performed two types of experiments in order to analyze the force model and rendering algorithm for 6-DoF haptic texture rendering. On the one hand, they performed offline experiments to analyze 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].

36.5.1 Comparison with Perceptual Studies

As mentioned in §34.2, 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 analyze the vibratory motion induced by the force model. The virtual experiments required the simulation of probe-plate interaction as well as human dynamics.

Description of Offline Experiments

The spherical probe is modeled as a circular disk of diameter D and the textured plate as a sinusoidal curve, as shown in Figure 179. 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.

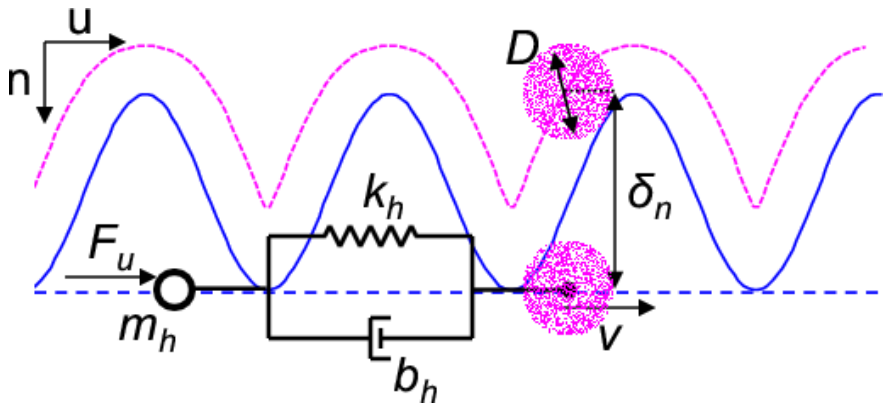


Figure 179: **Model of Probe-Surface Interaction and Grasping Dynamics.** *A disk moves on a sinusoidal texture at constant speed v while dragging a mass m_h. A texture force F_u, based on penetration depth $\delta_{\mathbf{n}}$, is applied to the mass.*

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

$$F_n = -k\delta_{\mathbf{n}}, \tag{61}$$

$$F_u = -k\delta_{\mathbf{n}}\frac{d\delta_{\mathbf{n}}}{du}. \tag{62}$$

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:

$$F_u = F_n\frac{d\delta_{\mathbf{n}}}{du}. \tag{63}$$

Human dynamics are modeled as a system composed of mass m_h, spring k_h, and damper b_h [HC02]. The mass is linked through the spring and damper to a point moving at constant speed v on the textured surface. The dragging force imposed by the point accounts for the influence of exploration speed, which is a factor analyzed by Lederman et al. [LKHR99]. Figure 179 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:

$$m_h\frac{d^2u}{dt^2} = k_h(vt-u) + b_h\left(v-\frac{du}{dt}\right) - F_u. \tag{64}$$

The experiments summarized 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. 64. 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 180 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 181 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.

Effects of Exploratory Speed

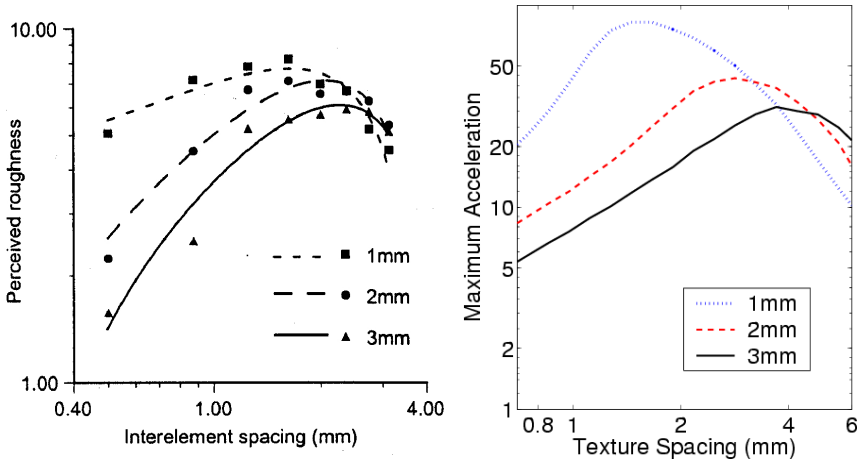


Figure 180: **Effects of Probe Diameter.** *Left: results of psychophysics studies by Lederman et al. [2000] (printed with permission of ASME and authors); Right: simulation results using a novel force model.*

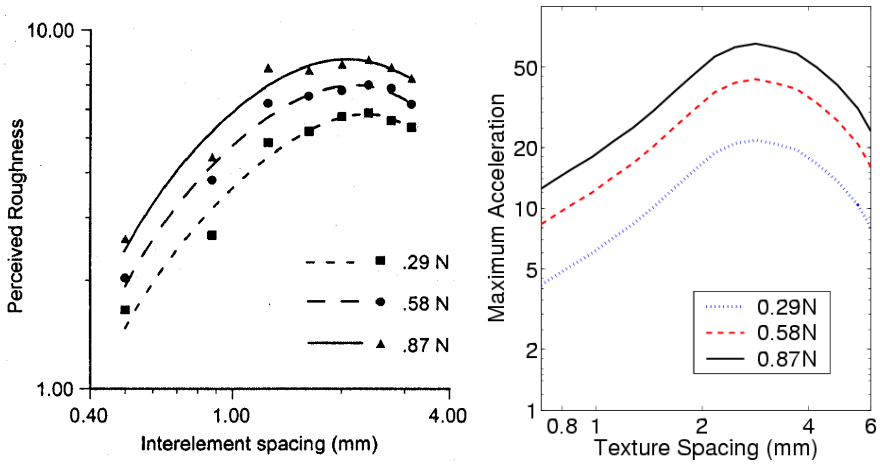


Figure 181: **Effects of Applied Force.** *Left: results of psychophysics studies by Lederman et al. [2000] (printed with permission of ASME and authors); Right: simulation results using a novel force model.*

Figure 182 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 §36.3.2 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.

36.5.2 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

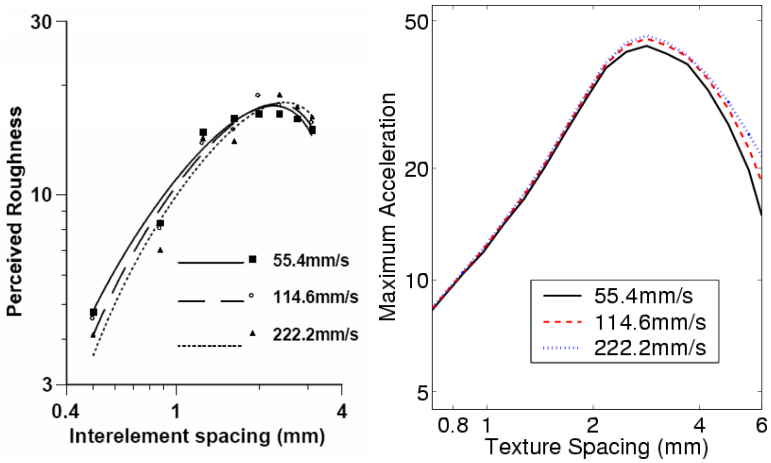


Figure 182: **Effects of Exploratory Speed.** *Left: results of psychophysics studies by Lederman et al. [1999] (printed with permission of Haptics-e and authors); Right: simulation results using a novel force model.*

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 §36.2.2, 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 §35. 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 §35.3.3).

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 stabilizing intermediate representation. In this way the experiments do not get distorted by the use of intermediate representations, and the analysis can focus on the performance of the force model and the rendering algorithm. 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

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).*

The models shown in Figure 183 were used for the experiments on conveyance of roughness. The performance tests were executed on the models shown in Figures 184 and 185. The complexities of the full-resolution textured models and their coarse resolution approximations are

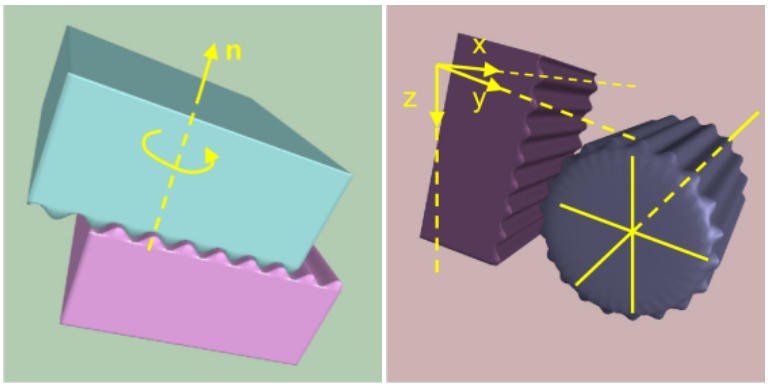


Figure 183: **Benchmark Models for Experiments on Conveyance of Roughness.** *Left (a): textured blocks; Right (b): block and gear.*

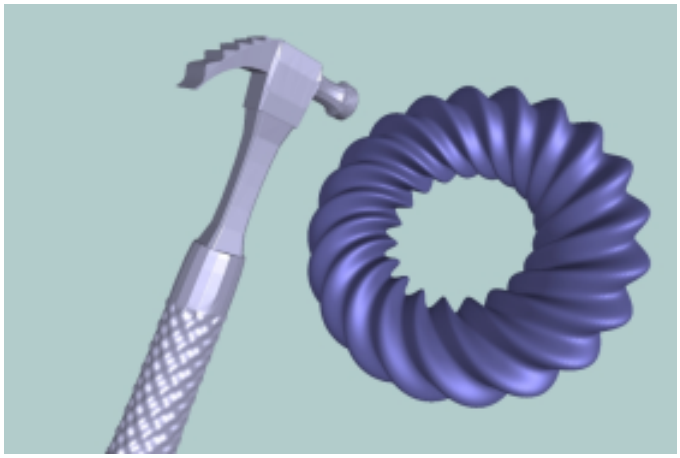


Figure 184: **Textured Hammer and Helicoidal Torus.** *Benchmark scenario for performance tests.*

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 analyzing force and motion characteristics in the simulations.

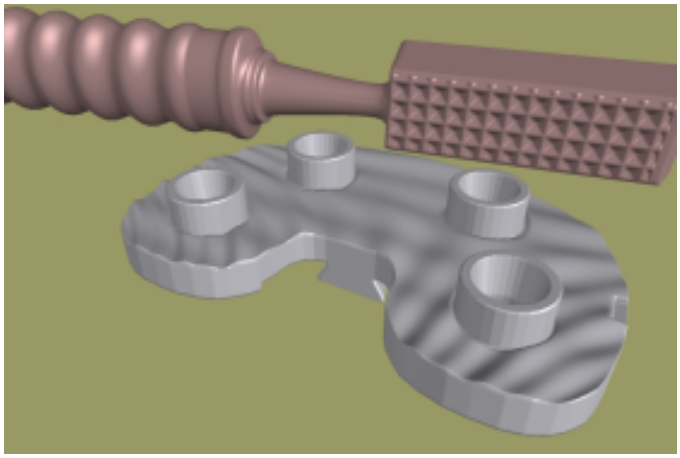


Figure 185: **File and CAD Part.** *Benchmark scenario for performance tests.*

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 183-b. Figure 186 shows the position of the block and the force exerted on it during $1,500$ frames of interactive simulation (approx. 3 seconds).

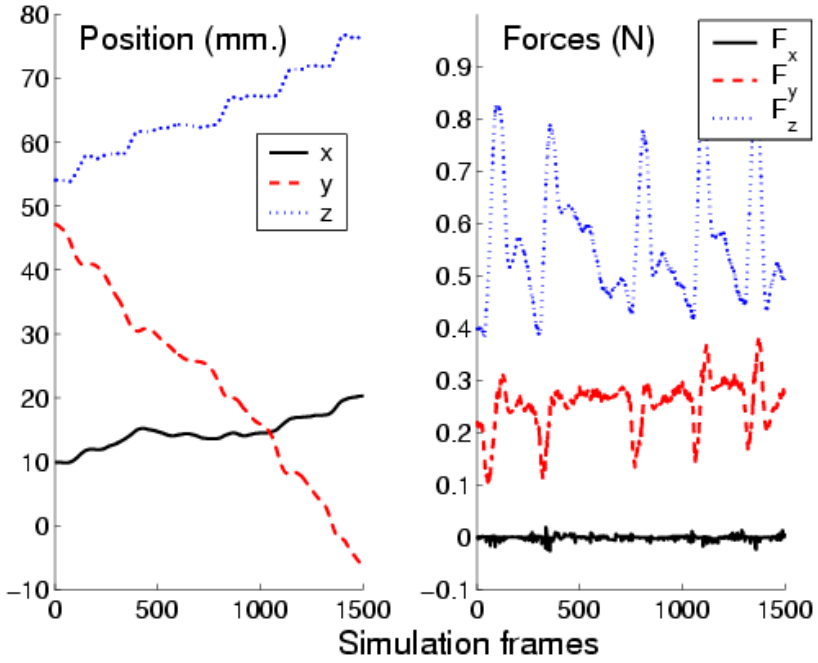


Figure 186: **Roughness under Translation.** *Position and force profiles generated while translating the model of a textured block in contact with a gear model, as shown in Figure 183-b. Notice the staircase-like motion in z, and the correlation between force and position changes.*

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

Conveyance of Roughness under Rotation

Two identical striped blocks were placed interlocking each other, as shown in Figure 183-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 187 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 §36.3 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 185 and 184).

In particular, Figure 188 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 §36.4.2). 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

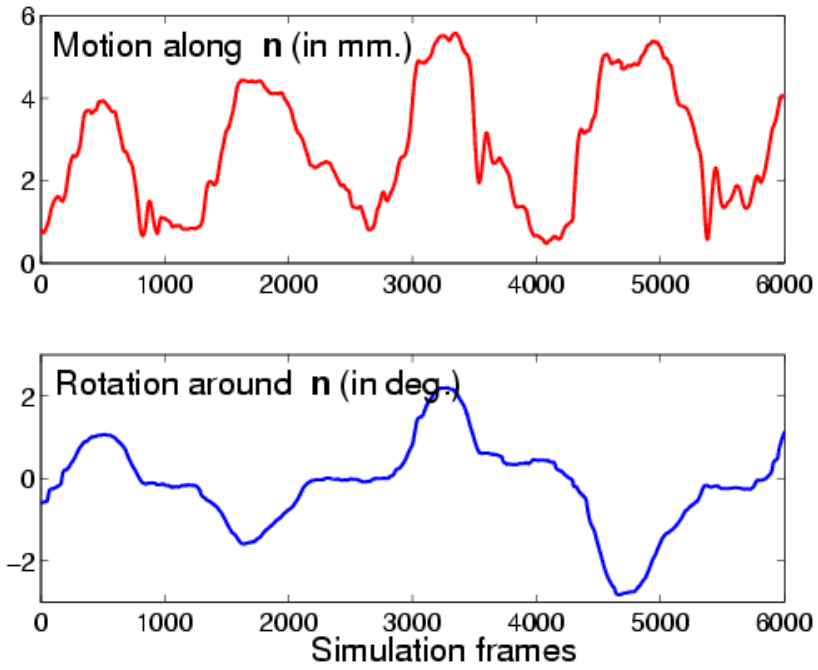


Figure 187: **Roughness under Rotation.** *Motion profile obtained by rotating one textured block on top of another one, as depicted in Figure 183-a. Notice the translation induced by the interaction of ridges during the rotational motion.*

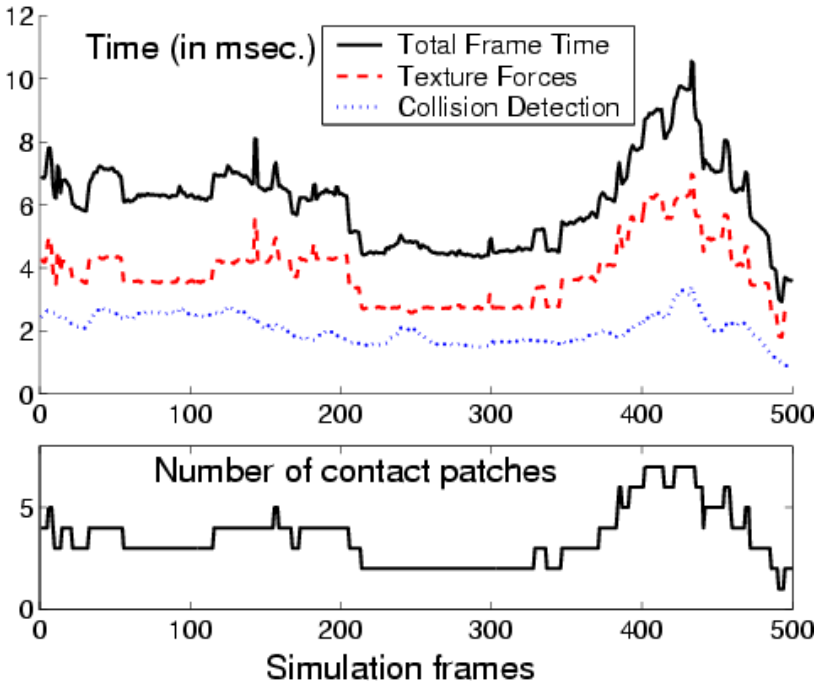


Figure 188: **Timings.** *Performance analysis and number of clustered contact patches during* 500 *simulation frames of a file model scraping a CAD part, as shown in Figure 185. In this complex contact scenario the haptic frame rate varies between 100Hz and 200Hz.*

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 188 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.

36.6 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.

36.6.1 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 §36.5, in practice the force model generates forces that create a realistic perception of roughness for object-object interaction; however, one essential limitation of penalty-based methods and impedance-type haptic devices (such as the 6-DoF PHANTOM 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 [CT03] 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.

36.6.2 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.

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 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.

A Appendix A: Implementation of the Dormand-Prince Method

We chose to use a Runge-Kutta method to solve equation 13. Expressing such equation in components we get:

$$\frac{d}{dl}\left(n_\lambda \frac{dx_j}{dl}\right) = \frac{\partial n}{\partial x_j} \tag{65}$$

with $j = 1,2,3$. It can be rewritten as:

$$n_\lambda \frac{d^2x_j}{dl^2} + \frac{dn_\lambda}{dl}\frac{dx_j}{dl} = \frac{\partial n}{\partial x_j} \tag{66}$$

$$\frac{d^2x_j}{dl^2} = \frac{1}{n_\lambda}\left(\frac{\partial n_\lambda}{\partial x_j} - \frac{dn_\lambda}{dl}\frac{dx_j}{dl}\right) \tag{67}$$

Changing the variable

$$y_j = \frac{dx_j}{dl} \tag{68}$$

we obtain the following equation:

$$y_j' = \frac{1}{n_\lambda}\left(\frac{\partial n_\lambda}{\partial x_j} - \frac{dn_\lambda}{dl}y_j\right) \tag{69}$$

with

$$\frac{dn_\lambda}{dl} = \frac{dn_\lambda}{dx_j}\frac{dx_j}{dl} \tag{70}$$

From equation 68 we have:

$$x_j' = y_j \tag{71}$$

Equations 69 and 71 define a system where x_j represents the position and y_j represents the velocity at a given point in the trajectory, which can be written in matrix form as:

$$\begin{pmatrix} x_j \\ y_j \end{pmatrix}' = \begin{pmatrix} y_j \\ \frac{1}{n_\lambda}\left(\frac{\partial n_\lambda}{\partial x_j} - \frac{dn_\lambda}{dl}y_j\right) \end{pmatrix} \tag{72}$$

This equation 72 has the form

$$Y' = f(l, Y) \tag{73}$$

which defines an Initial Value Problem with $Y(0) = \alpha$. The change of variable in equation 68 allows us to solve this problem by applying the embedded Runge-Kutta method RK5(4)7M from the Dormand-Prince family of methods. It has been chosen because it is an embedded method, which implies variable step and error control, and also because its convergence is much faster than other methods.

The RK5(4)7M obtains an approximate solution of Y in several points of the interval $[A, B]$, and then interpolates for the rest of the points. All Runge-Kutta methods are characterized by a table of coefficients:

c_1	0			
c_2	a_{21}	...	0	
...	...	...	...	0
c_s	a_{s1}	...	a_{ss-1}	0
	b_1	b_2	...	b_s
	$\hat{b}_1$	$\hat{b}_2$	...	$\hat{b}_s$

whereas for the chosen RK5(4)7M method these coefficients are:

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$-\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
	$\frac{5179}{56700}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$
	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0

To obtain the different trial points (x_n, y_n) which represent the approximation to the solution, we first need to obtain the k_n^i coefficients from the following equation:

$$k_n^i = f\left(x_n + h_n c_i, y_n + \sum_{j=1}^{i-1} a_{ij} k_n^j\right) \tag{74}$$

where h_n is the step size in the iteration n and a_{ij} y c_i are obtained from the table of coefficients. Given the coordinates (x_n, y_n) of a point, the coordinates (x_{n+1}, y_{n+1}) of the next point are obtained as

$$x_{n+1} = x_n + h_n \tag{75}$$

$$y_{n+1} = y_n + h_n \sum_{i=1}^{s} b_i k_n^i \tag{76}$$

where b_i are given by the coefficients table and k_n^i is obtained using equation 74. The other solution of the embedded pair is obtained as:

$$\hat{y}_{n+1} = y_n + h_n \sum_{i=1}^{s} \hat{b}_i k_n^i \tag{77}$$

The error estimation E_n for each step is then computed as:

$$E_n = y_n - \hat{y}_n \tag{78}$$

If this error is greater than a given tolerance tol, the step is modified and the solutions recalculated. If the error is less than such tolerance, the next point is calculated while also varying the step. The new step is calculated as follows:

$$h_{n+1} = h_n \left(\frac{tol}{\|E_n\|}\right)^{\left(\frac{1}{p}\right)} \tag{79}$$

Given the numerical precision used and the round error when implementing the method, it could be that the error $\|E_n\|$ becomes zero; values of $\|E_n\|$ less than a given ε must therefore be treated differently. In this implementation we chose to multiply the step times ten when this happens.

As it has been stated in this chapter, the functions and formulae used to derive our atmospheric profiles are based on measured data, or a comparison between predicted values and real data has been performed. We have borrowed extensively from scientific literature in other areas of research such as optics or meteorology, in order to ensure the predictability of the model.

To validate the resolution method, we have tested the implementation using a similar scheme than the proposed in [SGM*05], consisting on an empty scene where the index of refraction varies according to the equation $n = 1 + ky$, with y representing height, and k varying from -0.1 to 0.1. This distribution of n can be solved analytically, so we can measure the numerical error against the exact solution. The analytical and numerical solutions are compared shooting a ray into the scene and following it until it travels 100 distance units. We also Analise the results varying the error tolerance of the method. Figure 189 shows the error of the Dormand-Prince RK5(4)7M method as the tolerance is reduced, along with the time it takes to compute the solution. As it can be seen, error tolerances in the range of 10^{-8} to 10^{-12} yield good results without much of a time penalty. Error tolerances beyond 10^{-14} start increasing rendering times considerably. The validation tests were run on a P4 at 2.8 GHz and 1 GB RAM.

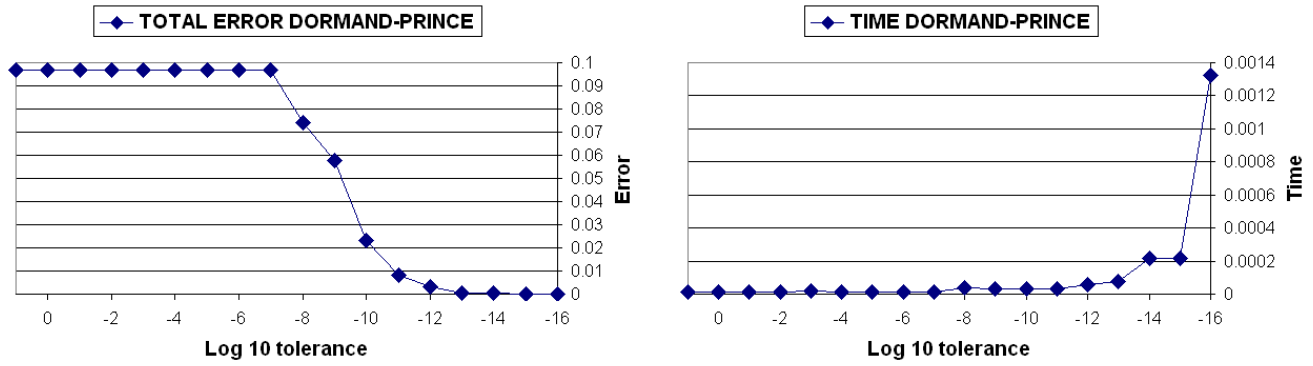


Figure 189: Error and rendering time (secs.) as functions of the error tolerance in the Dormand-Prince RK5(4)7M method for a test scene.

In terms of convergence, the chosen Dormand-Prince method is orders of magnitude faster than simpler numerical resolution methods like Euler.

B Appendix B: Fluorescence in Underwater Environments

The quantum efficiency Γ (as well as its related functions $f(x,\lambda_i \to \lambda)$ and $\eta(x,\lambda_i \to \lambda)$) depends on both the medium and the type of inelastic event considered, and it can be seen as a re-radiation matrix. Oceanographers have measured this matrix in different places of the world, and for this work we chose to use the data from the Gulf of Mexico [Mob94].

For chlorophyll, only the wavelengths between 370 and 690 nm. can trigger fluorescence. That can be modeled as a dimensionless function $g^c(\lambda_i)$ so that:

$$g^c(\lambda_i) \equiv\to \begin{cases} 1 & \text{if } 370 \leq \lambda_i \leq 690 \text{ nm.} \\ 0 & \text{otherwise} \end{cases} \tag{80}$$

Using equation 81, the relationship between the wavelength redistribution function $f(x,\lambda_i \to \lambda)$ and the spectral quantum efficiency function $\eta(x,\lambda_i \to \lambda)$ is:

$$f^c(x,\lambda_i \to \lambda) = \eta^c(x,\lambda_i \to \lambda)\frac{\lambda_i}{\lambda} \equiv \Gamma^c g^c(\lambda_i) h^c(\lambda)\frac{\lambda_i}{\lambda} \tag{81}$$

where $h^c(\lambda)$ is the fluorescence emission function for chlorophyll per unit wavelength, and can be approximated by a Gaussian:

$$h^c = \frac{1}{\sqrt{2\pi}\sigma^c} \exp\left[-\frac{(\lambda-\lambda_0^c)^2}{2(\sigma^c)^2}\right]. \tag{82}$$

where $\lambda_0^c = 685nm.$ is the wavelength of maximum emission and $\sigma^c = 10.6nm.$ represents the standard deviation. The phase function for chlorophyll-induced fluorescence is isotropic.

For fluorescence owed to high concentrations of yellow matter, Hawes [Haw92] concluded that its quantum efficiency $\Gamma^y(\lambda_i)$ was independent of the excitation wavelength λ_i, with $\Gamma^y(\lambda_i)$ values between 0.005 and 0.025. The proposed formula to describe the fluorescence efficiency function for yellow matter is:

$$\eta^Y(x,\lambda_i \to \lambda) = A_0(\lambda_i) \exp\left(-\left(\frac{\frac{1}{\lambda} - \frac{A_1}{\lambda_i} - B_1}{0.6\left(\frac{A_2}{\lambda_i} + B_2\right)}\right)\right) \tag{83}$$

where $A_0, A_1, A_2, B_1 y B_2$ are empirical parameters which values depend on the specific composition of yellow matter. A_1 and A_2 are dimensionless, whereas the rest represent nm^{-1}.

References

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.

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).

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).

ALHALABI M. O., HORIGUCHI S.: Network latency issue in cooperative shared haptic virtual environment. In *SPIE Third International Conference on Virtual Reality and Its Application in Industry* (April 2003), Pan Z., Shi J., (Eds.), vol. 4756, pp. 199–205.

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

AMDAHL G. M.: Validity of the single-processor approach to achieving large scale computing capabilities. In *AFIPS* (Atlantic City, Apr. 1967), vol. 30, AFIPS Press, Reston, Va.

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

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.

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).

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.

BACKUS J.: Can programming be liberated from the von Neumann style functional style and its algebra of programs. *Commun. ACM 21*, 8 (1978), 613–641.

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

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.

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.

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.

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

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

BURDEN R. L., FAIRES J.: *Numerical Analysis*, fourth ed. PWS-Kent Publishing Co., Boston, 1988.

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

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

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.

BASDOGAN C., HO C., 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.

BASDOGAN C., HO C., 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.

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.

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.

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

BRIDAULT F., LEBLOND M., ROUSSELLE F., RENAUD C.: Enhanced illumination of reconstructed dynamic environments using a real-time flame model. In *Afrigraph 2006* (2006), ACM Press.

BOLIN M. R., MEYER G. W.: A frequency based ray tracer. In *SIGGRAPH* (1995), pp. 409–418.

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

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

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

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.

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

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.

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

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

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

BERGER M., TROUT T., LEVIT N.: Ray tracing mirages. *IEEE Computer Graphics and Applications 10*, 3 (May 1990), 36–41.

BURKS A. W.: Programming and structural changes in parallel computers. In *Conpar* (Berlin, 1981), Händler W., (Ed.), Springer, pp. 1–24.

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

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

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

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

BORN M., WOLF E.: *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light*. Cambridge University Press, 2002.

BEEHAREE A. K., WEST A. J., HUBBOLD R. J.: 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.

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

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

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

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.

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.

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).

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.

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

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.

CHAIGNE A., DOUTAUT V.: Numerical simulations of xylophones. i. time domain modeling of the vibrating bars. *J. Acoust. Soc. Am. 101*, 1 (1997), 539–557.

CHALMERS A., DEVLIN K.: Recreating the past. In *SIGGRAPH 2002 Course* (2002), ACM SIGGRAPH.

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.

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

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

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.

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

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.

CHALMERS A. G.: Occam - the language for educating future parallel programmers? *Microprocessing and Microprogramming 24* (1988), 757–760.

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.

CIDDOR P.: Refractive index of air: new equations for the visible and near infrared. *Applied Optics 35*, 9 (1996), 1566–1573.

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.

CHUNG J. Y., LIU J. W. S., LIN K. J.: Scheduling real-time, periodic jobs using imprecise results. In *Proc. IEEE RTS* (1987).

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

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

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

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.

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

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.

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.

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).

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

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.

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

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.

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.

DEVLIN K., CHALMERS A., BROWN D.: Predictive lighting and perception in archaeological representations. In *NESCO "World Heritage in the Digital Age" 30th Anniversary Digital Congress* (October 2002), UNESCO World Heritage Centre.

DEUSEN O., EBERT D., FEDWIK R., MUSGRAVE K., PRUSINKIEWICZ P., ROBLE D., STAM J., J T.: The elements of nature: interactive and realistic techniques. In *SIGGRAPH 2004 Course 31* (2004).

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

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

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

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.

DONGARRA J. J.: *Performance of Various Computers Using Standard Linear Equations Software (LINPACK Benchmark Report)*. Tech. rep., University of Tennessee, Knoxville, TN, USA, 2005.

DORMAND J., PRINCE P.: A family of embeded runge-kutta formulae. *Journal of Computational and Applied Mathematics 6(1)* (1980), 19–26.

DUMONT R., PELLACINI F., FERWERDA J. A.: Perceptually-driven decision theory for interactive realistic rendering. *ACM Trans. Graph. 22*, 2 (2003), 152–181.

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.

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

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

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.

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.

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.

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

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

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.

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.

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.

FOUAD H., BALLAS J., HAHN J.: Perceptually based scheduling algorithms for real-time synthesis of complex sonic environments. In *Proc. Int. Conf. Auditory Display* (1997).

FLORENS J. L., CADOZ C.: The physical model: modeling and simulating the instrumental universe. In *Represenations of Musical Signals*, Poli G. D., Piccialli A.,, Roads C., (Eds.). MIT Press, Cambridge, MA, USA, 1991, pp. 227–268.

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

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

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).

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

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.

FERWERDA J. A., PATTANAIK S. N., SHIRLEY P., GREENBERG D. P.: A model of visual masking for computer graphics. *Computer Graphics 31*, Annual Conference Series (1997), 143–152.

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

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

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

GUTIERREZ D., ANSON O., MUNOZ A., SERON F.: Perception-based rendering: Eyes wide bleached. In *EG '05: Short Papers Proceedings of Eurographics* (2005), Dingliana J., Ganovelli F., (Eds.), The Eurographics Association and The Image Synthesis Group, pp. 49–52.

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

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

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).

GLADSTONE J. H., DALE J.: On the influence of temperature on the refraction of light. *Phil. Trans. 148* (1858), 887.

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

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.

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

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

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

GLENCROSS M., HUBBOLD R. J., 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.

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.

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.

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

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.

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

GLASSNER A. S.: A model for fluorescence and phosphorescence. In *Photorealistic Rendering Techniques* (1995), Sakas P. S. G., Müller S., (Eds.), Eurographics, Springer-Verlag Berlin Heidelberg New York, pp. 60–70.

GLASSNER A. S.: *Principles of Digital Image Synthesis*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995.

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

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.

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

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

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.

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

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.

GUTIERREZ D., MUNOZ A., ANSON O., SERÓN F. J.: Non-linear volume photon mapping. In *Proc. of the Eurographics Symposium on Rendering Techniques, Konstanz, Germany, June 29 - July 1, 2005* (2005), pp. 291–300.

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

GLENCROSS M., OTADUY M. A., CHALMERS A. G.: Interaction in distributed virtual environments. Eurographics Tutorial, August 2005.

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

GRÖLLER E.: Nonlinear ray tracing: visualizing strange worlds. *The Visual Computer 11*, 5 (1995), 263–276.

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.

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

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

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

GUSTAFSON J. L.: Re-evaluating Amdahl's law. *Commun. ACM 31*, 5 (May 1988), 532–533.

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

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

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

HARTREE D. R.: The ENIAC, an electronic computing machine. *Nature 158* (1946), 500–506.

HARRIS M. J.: 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.

HAWES S.: *Quantum fluorescence efficiencies of marine fulvic and humid acids*. PhD thesis, Dept. of Marince Science, Univ. of South Florida, 1992.

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.

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.

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

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.

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.

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

HEIDORN K. C.: http://www.islandnet.com/ see/weather/history/artmirge.htm, december 2005.

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.

HUDSON T. C., HELSER A. T., SONNENWALD D. H., WHITTON M. C.: Managing collaboration in the nanomanipulator. *Presence: Teleoperators and Virtual Environments 13*, 2 (April 2004), 193–210.

HOCKNEY R. W., JESSHOPE C. R.: *Parallel Computers 2*. Adam Hilger, 1988.

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

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.

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).

HABER J., MYSZKOWSKI K., YAMAUCHI H., SEIDEL H.-P.: Perceptually Guided Corrective Splatting. In *Computer Graphics Forum, Proc. of Eurographics 2001* (Manchester, UK, September 2001), Chalmers A., Rhyne T.-M., (Eds.), vol. 20 of *Computer Graphics Forum*, Eurographics, Blackwell, pp. C142–C152.

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.

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

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.

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

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

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.

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.

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).

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

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

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.

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.

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.

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.

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

IRWIN J.: Full-spectral rendering of the earth's atmosphere using a physical model of rayleigh scattering. In *Proceedings of the 14th Annual Eurographics UK Conference* (1996), vol. 1, pp. 103–115.

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

JENSEN H. W., CHRISTENSEN P. H.: Efficient simulation of light transport in scenes with participating media using photon maps. In *SIGGRAPH 98 Conference Proceedings* (jul 1998), Cohen M., (Ed.), Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 311–320. ISBN 0-89791-999-8.

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

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

JAY C., HUBBOLD R. J.: 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.

JAY C., HUBBOLD R. J.: 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.

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

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).

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).

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

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

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

KAYE J.: Making scents: Aromatic output for HCI. *ACM Interactions 11*, 1 (2004), 48–61.

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

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

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.

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.

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.

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.

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

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.

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.

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.

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.

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.

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).

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.

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.

KRASTOV Y.: *Geometrical optics of inhomogeneous media*. Springer Verlag, 1990.

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.

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

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

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.

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.

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

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

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.

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

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).

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

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

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).

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).

LONGHURST P., LEDDA P., CHALMERS A.: Psychophysically based artistic techniques for increased perceived realism of virtual environments. In *Afrigraph 2003* (2003), ACM Press, pp. 123–131.

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.

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.

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.

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.

LARSON G. W., RUSHMEIER H., PIATKO C.: A Visibility Matching Tone Reproduction Operator for High Dynamic Range Scenes. *IEEE Transactions on Visualization and Computer Graphics 3*, 4 (Oct. 1997), 291–306.

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.

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.

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

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.

MENABREA L. F., AUGUSTA(TRANSLATOR) A.: Sketch of the Analytical Engine invented by Charles Babbage. In *Charles Babbage and his Calculating Engines*, Morrison P., Morrison E., (Eds.). Dover Publications, 1961.

MALAURIE J.: *Ultima Thule: Explorers and Natives in the Polar North*. W. W. Norton & Company, 2003.

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.

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

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

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

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.

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

MIRTICH B., CANNY J.: Impulse-based simulation of rigid bodies. In *1995 Symposium on Interactive 3D Graphics* (Apr. 1995), Hanrahan P., Winget J., (Eds.), ACM SIGGRAPH, pp. 181–188. ISBN 0-89791-736-7.

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.

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

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

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.

MARSH J., GLENCROSS M., PETTIFFER S., HUBBOLD R. J.: A robust network architecture supporting rich behaviour in collaborative interactive applications. *IEEE Transactions on Visualisation and Computer Graphics TVCG 12*, 3 (May 2006), 405–416.

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

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

MINNAERT M.: *The nature of light and color in the open air*. Dover, New York, 1954.

MINNAERT M.: *Light and Colour in the Outdoors*. Springer-Verlag, New York, 1993.

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.

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

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

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

MITCHELL D. P.: Generating antialiased images at low sampling densities. In *SIGGRAPH '87: Proc. of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1987), pp. 65–72.

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

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.

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

MOBLEY C.: *Light and water. Radiative transfer in natural waters*. Academic Press, Inc., 1994.

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.

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

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

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

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.

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

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).

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.

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.

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

MAY D., SHEPHERD R.: *Communicating Process Computers*. Inmos technical note 22, Inmos Ltd., Bristol, 1987.

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.

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

MCLAUGHLIN M., SUKHATME G., PENG W., ZHU W., PARKS J.: Performance and co-presence in heterogeneous haptic collaboration. In *IEEE Eleventh Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS)* (2003).

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.

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

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

MUSGRAVE F. K.: A note on ray tracing mirages. *IEEE Computer Graphics and Applications 10*, 6 (Nov. 1990), 10–12.

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.

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.

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.

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

MYSZKOWSKI K.: Perception-based global illumination, rendering, and animation techniques. In *SCCG '02: Proc. of the 18th spring conference on Computer graphics* (New York, NY, USA, 2002), ACM Press, pp. 13–24.

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.

The Nautical Almanac. Her Majesty's Nautical Almanac Office, London, and Nautical Almanac Office of the United States, Washington DC, 1st ed., 1958.

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).

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

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

O'BRIEN J. F., COOK P. R., ESSL G.: Synthesizing sounds from physically based motion. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM Press, pp. 529–536.

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

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.

OTADUY M. A., JAIN N., SUD A., LIN M. C.: Haptic display of interaction between textured models. *Proc. of IEEE Visualization* (2004), 297–304.

OTADUY M. A., LIN M. C.: CLODs: Dual hierarchies for multiresolution collision detection. *Eurographics Symposium on Geometry Processing* (2003), 94–101.

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.

OTADUY M. A., LIN M. C.: A perceptually-inspired force model for haptic texture rendering. *Proc. of Symposium APGV* (2004), 123–126.

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.

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).

O'SULLIVAN C., RADACH R., COLLINS S.: A model of collision perception for real-time animation. *Computer Animation and Simulation* (1999), pp. 67–76.

O'BRIEN J. F., SHEN C., GATCHALIAN C. M.: Synthesizing sounds from rigid-body simulations. In *The ACM SIGGRAPH 2002 Symposium on Computer Animation* (July 2002), ACM Press, pp. 175–181.

OWENS J.: Optical refractive index of air: dependence on pressure, temperature and composition. *Applied Optics 6*, 1 (1967), 51–59.

OUH-YOUNG M.: *Force Display in Molecular Docking*. PhD thesis, University of North Carolina, Computer Science Department, 1990.

PESHKIN M., COLGATE J. E.: Cobots. *Industrial Robot 26*, 5 (1999), 335–341.

POPE J., CHALMERS A. G.: Multi-sensory rendering: Combining graphics and acoustics. In *Seventh International Conference in Central Europe on Computer Graphics* (1999).

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.

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.

PETTIFER S. R.: *An operating environment for large scale virtual reality*. PhD thesis, The University of Manchester, 1999.

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.

POTTER K., JOHNSON D., COHEN E.: Height field haptics. *Proc. of Haptics Symposium* (2004).

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.

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.

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).

PEREZ F., PUEYO X., SILLION F. X.: Global illumination techniques for the simulation of participating media. In *Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering)* (New York, NY, 1997), Dorsey J., Slusallek P., (Eds.), Springer Wien, pp. 309–320.

PRIKRYL J.: *Radiosity methods driven by human perception*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, 2001.

PIFKO A. B., WINTER R.: Theory and application of finite element analysis to structural crash simulation. *Computers and Structures 13* (June 1981), 277–285.

PANTEL L., WOLF L. C.: On the impact of delay on real-time multiplayer games. In *Twelveth International Workshop on Network and Operating Systems Support for Digital Audio and Video* (Miami, 2002), pp. 23–29.

QUINLAN S.: Efficient distance computation between non-convex objects. In *Proceedings of International Conference on Robotics and Automation* (1994), pp. 3324–3329.

RACZKOWSKI J.: Visual simulation and animation of a laminar candle flame. In *International Conference on Image Processing and Computer Graphics* (1996), Poland.

ROUSSOS I., CHALMERS A.: High fidelity lighting of knossos. In *VAST 2003* (2003), ACM Press, pp. 47–56.

REDDY M.: *Perceptually Modulated Level of Detail for Virtual Environments (CST- 134-97)*. PhD thesis, University of Edinburgh, 1997.

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.

REYNOLDS C. W.: Flocks, herds, and schools: A distributed behavioral model. In *SIGGRAPH Computer Graphics* (1987), vol. 21, pp. 25–34.

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.

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.

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.

RIESZ R., KLEMMER E. T.: Subjective evaluation of delay and echo supressors in telephone communications. *Bell System Technical Journal 42* (November 1963), 2919–2933.

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).

RUSPINI D., KOLAROV K., KHATIB O.: The haptic display of complex graphical environments. *Proc. of ACM SIGGRAPH* (1997), 345–352.

RUSHMEIER H., LARSON G. W., PIATKO C., SANDERS P., RUST B.: Comparing real and synthetic images: Some ideas about metrics. In *Eurographics Rendering Workshop* (June 1995), pp. 82–91.

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.

ROCK I., VICTOR J.: Vision and touch: An experimentally created conflict between the two senses. *Science 143* (1964), pp. 594–596.

The Star Almanac for Land Surveyors. Her Majesty's Nautical Almanac Office, London, 1st ed., 1951.

SALISBURY J. K.: Making graphics physically tangible. *Communications of the ACM 42*, 8 (1999).

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.

SHEN X., BOGSANYI F., NI L., GEORGANAS N. D.: A heterogeneous scalable architecture for collaborative haptics environments. In *IEEE International Workshop on Haptic Audio and Visual Environments (HAVE)* (September 2003), pp. 113–118.

SIMONS D., CHABRIS C.: Gorillas in our midst: Sustained inattentional blindness for dynamic events. *Perception 28* (1999), 1059–1074.

SUNDSTEDT V., CHALMERS A. G., CATER K., DEBATTISTA K.: Top-down visual attention for efficient rendering of task related scenes. In *Vision, Modeling and Visualization* (2004).

SUNDSTEDT V., DEBATTISTA K., LONGHURST P., CHALMERS A. G., TROSCIANKO T.: Visual attention for efficient high-fidelity graphics. In *Spring Conference on Computer Graphics SCCG* (May 2005).

SEIDEL R.: Linear programming and convex hulls made easy. In *Proc. 6th Ann. ACM Conf. on Computational Geometry* (Berkeley, California, 1990), pp. 211–215.

STAM J., FIUME E.: Turbulent wind fields for gaseous phenomena. In *Computer Geraphics* (August 1993), vol. 9, pp. 369–376.

STOKES W. A., FERWERDA J. A., WALTER B., GREENBERG D. P.: Perceptual illumination components: a new approach to efficient, high quality global illumination rendering. *ACM Trans. Graph. 23*, 3 (2004), 742–749.

SHIRMOHAMMADI S., GEORGANAS N. D.: An end-to-end communication architecture for collaborative virtual environments. *Computer Networks 35*, 2-3 (2001), 351–367.

SANDER P. V., GU X., GORTLER S. J., HOPPE H., SNYDER J.: Silhouette clipping. *Proc. of ACM SIGGRAPH* (2000), 327–334.

SUNDSTEDT V., GUTIERREZ D., GOMEZ F., CHALMERS A.: Participating media for high-fidelity cultural heritage. In *Sixth International Symposium on Virtual Reality, Arcgaeology and Cultural Heritage (VAST 2005)* (2005), Eurographics.

SERON F., GUTIERREZ D., GUTIERREZ G., E. C.: Implementation of a method of curved ray tracing for inhomogeneous atmospheres. *Computers and Graphics 29(1)* (2005).

SERON F., GUTIERREZ D., MAGALLON J., FERRAGUT L., ASENSIO M.: Visualization of a wildland forest fire front. *The Visual Computer* (Aceptado, en fase de publicacion (2005)).

SHAW J.: Distributed legible city. In *IST 98 — Information Society Technologies Conference and Exhibition* (Vienna, Austria, 1998).

SHIMOGA K.: Finger force and touch feedback issues in dextrous manipulation. *NASA-CIRSSE International Conference on Inetelligent Robotic Systems for Space Exploration* (1992).

SAWATZKY H. L., LEHN W. H.: The arctic mirage and the early north atlantic. *Science Magazine 192* (June 1976), 1300–1305.

STAM J., LANGUÉNOU E.: Ray tracing in non-constant media. In *Eurographics Rendering Workshop 1996* (New York City, NY, June 1996), Pueyo X., Schröder P., (Eds.), Eurographics, Springer Wien, pp. 225–234. ISBN 3-211-82883-4.

SLATER M.: Measuring presence: A response to the witmer and singer presence questionnaire. *Presence Teleoperators and Virtual Environments 8*, 5 (1999), 560–565.

SEK A., MOORE B. C.: Frequency discrimination as a function of frequency, measured in several ways. *J. Acoust. Soc. Am. 97*, 4 (April 1995), 2479–2486.

SILLION F. X., PUECH C.: *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco, CA, USA, 1994.

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.

SALLNÄS E., 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.

SANDER P. V., SNYDER J., GORTLER S. J., HOPPE H.: Texture mapping progressive meshes. *Proc. of ACM SIGGRAPH* (2001), 409–416.

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.

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.

STAM J.: Interacting with smoke and fire in real-time. *Communications of the ACM 43*, 7 (July 2000), 76–83.

SLATER M., USOH M.: *An Experimental Exploration of Presence in Virtual Environments*. Tech. Rep. 689, Department of Computer Science, University College London, 1993.

SUTHERLAND I. E.: The ultimate display. In *IFIP Congress*, (1965), pp. 506–508.

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).

SUNDSTEDT V. CHALMERS A. M. P.: High fidelity reconstruction of the ancient egyptian temple of kalabsha. In *AFRIGRAPH 2004* (2004), ACM SIGGRAPH.

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.

SINGHAL S., ZYDA M.: *Networked Virtual Environments Design and Implementation*. ACM Press SIGGRAPH Series Addison Wesley, 1999.

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.

TERZOPOULOS D., FLEISCHER K.: Modelling in-elastic deformation: Viscoelasticity, plasticity, fracture. In *SIGGRAPH* (1988), ACM Press, pp. 269–278.

TERZOPOULOS D., FLEISCHER K.: Deformable models. *Th Visual Computer 4* (1998), 306–331.

TOLKIEN J. R. R., GORDON E. V., DAVIS N.: *Sir Gawain and the Green Knight*. Oxford University Press, USA, 1968.

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.

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.

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.

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.

USGPC: *U.S. Standard Atmosphere*. United State Government Printing Office, Washington, D.C., 1976.

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.

ULICNY B., THALMANN D.: Towards interactive real-time crowd behavior simulation. *Computer Graphics Forum 21*, 4 (December 2002), 767–775.

ULBRICHT C., WILKIE A., PURGATHOFER W.: Psychophysically based artistic techniques for increased perceived realism of virtual environments. In *EG2005 STAR* (2005), Eurographics, pp. 95–112.

VAN DEN DOEL K., KRY P. G., PAI D. K.: Foleyautomatic: physically-based sound effects for interactive simulation and animation. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM Press, pp. 537–544.

VAN DEN DOEL K., KNOTT D., PAI D. K.: Interactive simulation of complex audiovisual scenes. *Presence: Teleoper. Virtual Environ. 13*, 1 (2004), 99–111.

VAN DEN DOEL K., PAI D. K.: Synthesis of shape dependent sounds with physical modeling. In *Proceedings of the International Conference on Auditory Displays* (1996).

VAN DEN DOEL K., PAI D. K.: The sounds of physical shapes. *Presence 7*, 4 (1998), 382–395.

VAN DER WERF S. Y.: Ray tracing and refraction in the modified us1976 atmosphere. *Applied Optics 42*, 3 (January 2003), 354–366.

VAN DER WERF S. Y., KÖNNEN G. P., LEHN W. H.: Novaya zemlya effect and sunsets. *Applied Optics 42*, 3 (January 2003), 367–378.

VERNE J.: *Le Rayon-Vert*. 1882.

VINCE J.: *Virtual Reality Systems*. Acm Siggraph Books Series, ACM Press/Addison-Wesley Publishing Co, New York, USA, 1995.

V.K. B.: *Refraction Tables of the Pulkovo Observatory, 5th ed.* Nauka, Leningrad, 1985.

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.

Welding simulator. Online Document, May 2005. http://www.vrsim.net/Products/weldingsimmain.html.

WALLACE M.: Vrsim welding application. Personnal correspondance with CEO of VRSim, 2005.

WELCH G., BISHOP G., VICCI L., BRUMBACK S., KELLER K., COLUCCI D.: High-performance wide-area optical tracking - the HiBall tracking system. In *Presence: Teleoperators and Virtual Environments* (2001), vol. 10.

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.

WATSON B., FRIEDMAN A., MCGAFFEY A.: An evaluation of level of detail degradation in head-mounted display peripheries. *Presence: Teleoperators and Virtual Environments 6*, 6 (1997), 630–637.

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

WHITEHOUSE D. J.: *Handbook of Surface Metrology*. Institute of Physics Publishing, Bristol, 1994.

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.

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).

WAN M., MCNEELY W. A.: Quasi-static approximation for 6 degrees-of-freedom haptic rendering. *Proc. of IEEE Visualization* (2003), 257–262.

WITMER B. G., SINGER M. J.: Measuring presence in virtual environments: A presence questionnaire. *Presence Teleoperators and Virtual Environments 7*, 3 (1998), 225–240.

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.

WALD I., SLUSALLEK P., BENTHIN C., WAGNER M.: Interactive rendering with coherent ray tracing. *Computer Graphics Forum 20*, 3 (2001).

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.

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.

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.

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.

YEE H.: *Spatiotemporal Sensitivity and Visual Attention for Efficient Rendering of Dynamic Environments*. Master's thesis, Cornell University, 2000.

YOUNG A. T.: http://mintaka.sdsu.edu/gf/, september 2005.

YOUNG A.: Visualizing sunsets iii. visual adaptation and green flashes. *Journal of the Optical Society of America A 17* (2000), 2129–2139.

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.

YOON S., SALOMON B., LIN M. C., MANOCHA D.: Fast collision detection between massive models using dynamic simplification. *Eurographics Symposium on Geometry Processing* (2004).

ZEEBERG J. J.: *Climate and Glacial History: Of the Novaya Zemlya Archipelago, Russian Artic With Notes on the Region's History of Exploration*. Thela Thesis, 2001.

ZWICKER E., FASTL H.: *Psychoacoustics*. Springer-Verlag, Berlin, 1990.

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.

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).