SIGRAD2003

The Annual SIGRAD Conference
Special Theme – Real-Time Simulations
November 20–21, 2003
Umeå Unversity, Umeå, Sweden,

Conference Proceedings

organized by

Svenska Föreningen för Grafisk Databehandling,
Umeå University,
VRlab, Umeå University,
Oryx Simulations
and
HPC2N - High Performance Computing Center North

Edited by

Mark Ollila

Published for Svenska Föreningen för Grafisk
Databehandling (SIGRAD) by
Linköping University Electronic Press
Linköping, Sweden, 2003

Linköping Electronic Conference Proceedings, No. 10
Linköping University Electronic Press
Linköping, Sweden, 2003

ISBN 91-7373-797-6 (print)
ISSN 1650-3686 (print)
http://www.ep.liu.se/ecp/010/
ISSN 1650-3740 (online)

Print: UniTryck, Linköping, 2003

Table of Contents

Welcome to SIGRAD2003 in Umeå!

Following the success of last year's conference in Norrköping, SIGRAD is for the second time arranging the annual computer graphics conference as a two day event. This year the conference is hosted by Umeå University and the theme of the event is real time simulations for computers graphics. Papers accepted for presentation are primarily within this area of computer graphics. The conference is, however, as always, open to contributions from other fields of computer graphics and applications.

The goal of the SIGRAD2003 conference is to provide a Nordic (yet international) forum for communication of recent research and development results in computer graphics and to bring together experts for a fruitful exchange of ideas and discussion on future challenges. Another important goal for the conference is to promote the exchange of experiences from fundamental and applied computer graphics with commercial R&D.

This year we are proud to have as keynote speaker Jos Stam who is one of leading researchers in the field of fluid simulations for computer graphics applications. His work has played a crucial role in introducing physical simulation into the computer graphics community.

Visual and interactive simulations are becoming increasingly more important as enabling technologies for the computer games industry which strive for ultimate realism, and poses interesting research challenges for many years to come. We are pleased to have Johan Persson, the original creator of the highly successful computer game, Battlefield 1942, as an invited speaker. He will share with us the story of the making of the game.

The papers that will be presented at this year's conference range from full conference publications to presentations of work in progress. It is noteworthy that many of the papers are authored and presented by graduate students and in some cases even undergraduate students in the field. In this manner SIGRAD also provides an important forum for the new generation of researchers in the field who, in a relaxed setting, can present and discuss their work. We thus also hope that the conference will serve as source of inspiration for the all the undergraduate students that will attend. Computer graphics in Sweden is indeed going through an impressive expansion and it is our hope that the SIGRAD conference series will continue to act as a catalyst for this development.

We would like to express our gratitude and warm welcome to the keynote speakers, authors of contributed papers, and other participants. We would also like to thank our sponsors, VRlab, HPC2N, Umeå University Department of Computing Science, CINS, EC Structural funds, SIGRAD, DICE, ReachIn Technologies and Oryx Simulations.

We wish you a most pleasant stay in Umeå.

Anders Backman, organizing co-chair

Kenneth Holmlund, organizing co-chair

Anders Ynnerman, SIGRAD chairman

SIGRAD2003 Program Committee

SIGRAD2003 consisted of experts in the field of computer graphics from all over Sweden. We thank them for their comments and reviews.

Tomas Akenine Möller, Chalmers
Niclas Börlin, Umeå University
Mike Connell, Chalmers
Matthew Cooper, Linköping University, ITN
Mark Dieckman, Linköping University, ITN
Mikael Jern, Linköping University, ITN
Lars Kjelldahl, KTH
Claude Lacoursière, Umeå University and CMLabs
Haibo Li, Umeå University
Mark Ollila, Linköping University, ITN
Stefan Seipel, Uppsala University
Odd Tullberg, Chalmers
Anders Ynnerman, Linköping University, ITN

SIGRAD2003 Organising Committee
Anders Backman, co-chair
Kenneth Holmlund, co-chair
Marcus Maxhall, conference dinner
Lena Hellman, coordination and administration
Niklas Edmundsson
Markus Mårtensson
Claude Lacoursière
Daniel Sjölie
Peter Sunna
Odd Tullberg
Charlotte Åkerlund

SIGRAD Board for 2003
Anders Ynnerman, Chair
Mark Ollila, Vice Chair
Lars Kjelldahl, Treasurer
Anders Backman, Secretary
Kai-Mikael Jää-Aro, Member
Charlotte Åkerlund, Member
Kenneth Holmlund, Substitute
Björn Kruse, Substitute
Åke Thurée, Substitute
Odd Tullberg, Substitute
Harald Tägnfors, Substitute
Örjan Vretblad, Substitute

The making of Battlefield 1942

Johan Persson

Digital Illusions

Abstract

Come and listen to the story of how Battlefield 1942 was made. Starting with the first steps of coming up with a game-concept and persuading a publisher to sign a deal. Moving on to the huge efforts, complex process and the hurdles that needed to be overcome in production, and ending with the final stages of Alpha and Beta-testing.

Speaker Bio

Johan Persson has a masters-degree in "Computer science and engineering" from KTH. He was one of the co-founders of Refraction Games, the studio behind "Codename Eagle". Today this studio is Digital Illusions' Stockholm office. He had the original concept and was the lead-programmer and physics-programmer for "Battlefield 1942", a game that has so far sold well beyond a million copies world-wide. Johan Persson is currently creative director at Digital Illusions.

Real-Time Simulations for Animation and Control of Physical Phenomena

Jos Stam

Alias

Abstract

In this talk we will present some of my research done at Alias Systems in real-time simulations of physical phenomena. In particular, we will describe a stable solution of fluid flow. But real-time cloth, hair, caustics and diffraction effects will also be covered. We will then show how these fast simulations can be used to control physical systems to achieve particular effects. For example, we will describe my recent work with Treuille et al from the University of Washington in controlling smoke animations using key frames.

Speaker Bio

Jos Stam is one of Alias's Research Scientists. His main interests are in the areas of natural phenomena modeling, such as smoke and fire, illumination models, physics-based animation and subdivision surfaces. In general, he is interested in any research that involves math and cool pictures.

He was born in the Netherlands and educated in Geneva, Switzerland where he studied pure mathematics. After moving to Canada, Jos received a PhD in Computer Science from the University of Toronto in 1995.

In 1997, Jos joined Alias after spending two years in Europe in various research labs. Throughout his studies and during his career with Alias Jos has published several papers at SIGGRAPH and elsewhere that address various areas of computer graphics.

A survey of some perceptual features for computer graphics and visualization

Lars Kjelldahl, KTH, Stockholm, Sweden
lassekj@nada.kth.se

Abstract

Some high level and some low level features of perception related to computer graphics and visualization are discussed in this survey paper. The need for perception in computer graphics and visualization will be important for the future of the area.

CR Categories: A1 [Survey], I.3.m [Computer Graphics miscellaneous]

Keywords: perception, computer graphics, visualization

1 Background

Computer graphics and visualization often have had a focus on speed and memory. The pictorial result has been accepted or rejected after a quick review by the author. During the last few years more interest has been oriented towards the quality of the image and how the methods can be adjusted in order to obtain a good or better result without unnecessary sacrifice of calculation efforts. In some applications one may accept lower quality in order to achieve speed. The limitations of the human visual system are used as a guide. We need knowledge on how scenes and visual information can be displayed so that its use is optimized for the observer. Details that can't be perceived by the observer should not be calculated and displayed.
From the beginning of computer graphics and visualization there has always been an influence from perception. One example of this is the experimental way of working that has been used. Fast visual inspection is in fact a way of using perceptual evaluation to see what methods that work.
From a low level hardware point of view perception has also been used. Integration features in the human visual system are used in CRT displays. Phosphor dots on the screen are fused when viewed on some distance from the screen. Another example is given by temporal variation (animation) of intensity on a screen, where integration in the human visual system makes the viewer perceive a stable image.

Investigation of perception and its relation to the physical world is much older than computer graphics. New is that we are able to manipulate the objects and the data in a much more flexible way than before which also means that we need to know more details on what effects we get by adding and removing different features in an image.

The diagram in figure 1 is confusing to many viewers. This is due to the choice of camera position, which most viewers would expect to be from above. A comment that the camera position is from below clarifies the presentation.

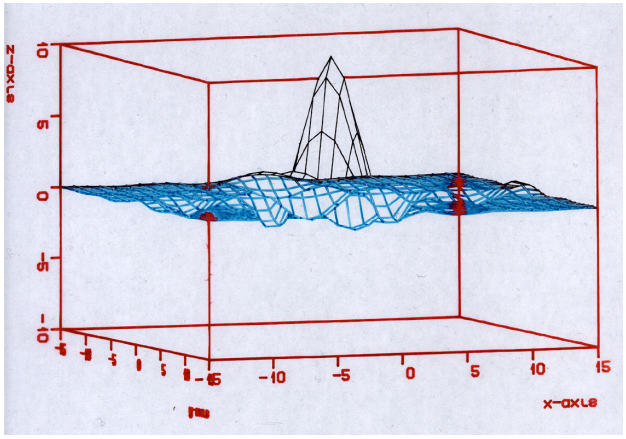


Figure 1: Graphical presentation of a mathematical function

The interest for perception in graphics and visualization has during recent years been manifested by articles in magazines, special issues of journals, tutorials at conferences etc. Two examples are the short presentation by Mahoney [2001] and the SIGGRAPH tutorial on experimental studies for computer graphics given by Ferwerda et.al. [2002].

The area of visual perception for computer graphics and visualization is interdisciplinary in its nature. Visual perception is important and related knowledge can be collected from people from many disciplines, e.g. perception researchers using psychophysical methods, neuroscientists, artificial network researchers, vision researchers, graphics and visualization researchers, graphics designers, artists and last but not least from application oriented persons.

Perceptual results are often given by psychophysical experimental studies. These results often have a quantitative character. Much research are more oriented towards qualitative investigations which is needed to understand fundamental mechanisms and performances in human-computer interaction. This would imply that perception is less interesting. According to our view, investigations and experimental studies on visual perception have a goal of achieving a pattern of understanding which is a qualitative result although a single experiment seems to be quantitative in nature.

2 Reasons for using perception

There are several different reasons for taking into account the perceptual aspects in graphics and visualization. Below is a list of questions where perception may contribute to an answer. For some questions some hints are included.
• how can we adjust rendering calculations to make the images more realistic?
• how can we present information and data in a way that make patterns in the data easier to detect?
Healey et al. [1996, 1998, 1999, 2002] discuss these questions.

• how can we avoid mistakes in the human understanding of an image?
Figure 1 gives an example of a possible misunderstanding.
• how can we emphasize the important parts of an image and how can we suppress the less important parts?
• how can we adjust the presentation/image to be more pleasing to the user?
• how can we avoid bad colour choices and colour combinations in images and presentations?
• how can we in scientific visualization find a good mapping between data dimensions and visual dimensions?
• how can we adjust our algorithms to avoid making unnecessary calculations of features that the human visual system will not perceive?
• how can limitations in hardware be taken care of?
Antialiasing and tone mapping are used here.

3 Application domain

The application and the task that you are working with apparently influence the visual features that should be used. Low level tasks include target detection, boundary detection, counting and estimation. High level tasks include modeling, image presentation, scientific visualization and computer graphics.

Application oriented issues include:
• modeling: how can you help the user to see the patterns needed to reveal the 3D structure and the relations between objects in the 3D environment?
• image presentation: how much can we manipulate a picture and still comprehend the picture as being identical to the original?
• data visualization: how can you help the user to recognize or detect patterns, outstanding objects etc?
• human computer interaction: what role does the graphical presentation have for interaction?
• compression: how much can we remove and still comprehend what we need?

Different categories of researchers and practitioners are interested in perceptual aspects of pictures - they use different methods to find out about knowledge related to images. Healey and Enns [2002] discuss common features for visualizations and professional artistic paintings showing in an experiment that people judge some kind of visualization pictures as comparable to patintings. Non photorealistic rendering relies on artistic values for the quality.
[Horton 95] gives a list of possible blunders by designers. The possible blunders may deal with for instance cultural differences, i.e. we interpret images differently depending on our background related to gestures, icons and colour. Other examples deal with resolution and integrated organization of text and images.
Artists use different effects to enforce impressions and to transfer information to the viewer (user). For instance the Egypts drew important persons bigger than the unimportant ones.

Categories of researchers/users that are interested in perceptual aspects of images include:
• perception researchers using psychophysical methods
• neuroscientist
• artificial network researchers
• vision researchers
• graphics and visualization researchers
• graphics designers
• artists
• photographers

4 The human visual system

The eye is the registrating part of the visual system and it is therefore important to have some fundamental knowledge about the eye. Here we will point out some pertinent properties of the eye that may be used in graphics/visualization systems.
The resolution of the eye is often measured in cycles per degree (c/deg) and is dependent on contrast but has a limit of about 60 c/deg, when it drops to zero [Reddy 2001]. For special objects the resolution may however be somewhat better. Hyperacuity makes it possible for us to perceive certain stimuli that are 10 times smaller than the size of the cones. The reason for the extra resolution is probably due to differences between integration of signals from many receptors in the eye.

There are two different kinds of photoreceptors in the eye, i.e. the cones (located mainly in the neighborhood of the fovea) and the rods located outside the fovea. The rods have a spectral sensitivity maximum at 500 nm, while the cones are of three kinds with different maxima, 450 nm ("blue" or S for short wave length), 525 nm ("green" or M for medium wave length) and 555 nm ("red" or L for long wave length). There are approximately 120 million rods, and 6 million cones. The 120 million receptors transmit their signals to layers of cells in the retina that can e.g. perform primitive detection of circular areas based on illumination as well as movements.
The S cones seems to be a little different from the M and L cones [Mollon 2000]. The S cones are more rare, only 8% of the cones are of S type. The S cones also combine their signals through a special cell (bipolar cell) with 2-3 cones for each bipolar cell, which means that the S signals yield a poor spatial resolution. The last cell layer in the eye consist of the ganglion cells delivering their signals to the optical nerve containing about one million fibers. The optical nerve from the two eyes then cross each other before entering the visual cortex of the brain.

It should be observed that some people have deficiency in detecting differences between red and green. There is a relation between this deficiency and a property of the x cromosone. As a result of this more men (approximately 8%) than women (approximately 0.4%) have problems to detect pattern variations in red/green. Stilling [1877] was the first person to design pseudoisochromatic plates for detecting colour deficiency where the dots in the pattern randomly vary in size and lightness. These test pictures are similar to the way fruit and berries appear in nature. Mammals might actually have developed colour vision to detect fruit and berries in nature [Mollon 2000].

The signals from the three cone types are combined to three new types before they are sent through the optic nerve. Red (L) and green (M) are combined to luminance, red and green are combined to a ratio between those two colours and finally blue (S) and luminance are combined to a ratio between yellow and blue. The luminance signal is used later in the visual system for detecting edges etc which means that borders using only blue are hard to detect clearly. A problem with the yellow-blue signal is also that the number of blue cones are scarce in the central fovea making it hard to detect fine details in blue.

An important aspect of how we use the human visual system is the focus of the attention, which can be bottom-up/stimulus driven or top-down/goal directed.
The bottom-up features are also called preattentive features and are processed in parallel in the visual system in only about 200 ms. The top-down features usually include a linear search through the data which may make this feature substantially more time consuming and dependent of the number of data items. It should be observed that the visual system often cannot integrate multiple features preattentively and that the preattentive features may be dependent of the focus and interest of the observer. The preattentive features includes colour, shape, direction, texture etc. However also some more high-level features may be processed preattentively as for instance three-dimensional shape and orientation.

Research indicates that processing of visual information in the brain takes place in at least three different "pathways" [Rheingans, Landreth 95]:
• "blob-thin-stripe-V4", processes spatial distribution of colours
• "parvo-interblob-pale-stripe-V4", processes high-resolution shape information
• "magno-4B-thick-stripe", processes movement and stereoscopic depth.
This indicates for instance that continuous presentations are quite different from discrete presentations. Continuous presentations make the viewer focus on global aspects while discrete presentations make the viewer focus on shape details.

5 Investigation methods

Investigation methods for the use of inages include methods from different fields. We discuss a few of these methods here and use colour as an example.
Colour is usually the single most important surface property. Colour can be categorized in different ways. [McCann 2000] discusses four different levels:

Problem	Discipline	Model
scene	physics	colourimetry
appearance	psychophysics	sensation
recognition	AI	perception
aesthetics	fine arts	visualization

The first category models the response of the cones and rods at the retina of the eye and has physics as a base for the building of the concepts. The second category models responses but at the same time taking into account the human perceptual response. This discipline is called psychophysics and has a focus on how colours appear. The third category models the recognition of colour. It is related to our earlier experience. The forth category models emotion and is related to what artists and designers are doing.

Methods for evaluation of perceptual efficiency/quality often rely on some image quality metric. The construction of these metrics depend on what information the images are intended to communicate. An overview of image quality metrics is given in [Chalmers et al. 2000].

Most image metrics deal with low level features oriented towards photorealism. In [Shacked and Kischinski 2001] six "target" terms are given to control the quality of the image. The last of these terms is regarded as optional. It constraints the light coming from above and is as a matter of fact important for the interpretation of the shape, which can be seen in the examples in the paper.

Different kind of measures have been developed:
• comparison of actual photo and computer generated image
• comparison between several images such as in an iterative sequence of synthetic generated images
• user evaluation of the image
• evaluation of several images (gallery)
• some kind of expert evaluation of the image, by an artist, a designer, an application oriented person or some other person.

6 High level features

Our vision can be divided into different levels of perception. Low level perception deals with for instance how small details we can actually see (usually related to features computed in the eye). High level perception deals more with our understanding of the concepts in the scenes presented to us (usually related to features computed in the visual cortex of the brain). High level concepts use our experience of how the world usually is built and presented to us. Examples of properties that the human may use:
• rooms usually have 90 degree corners
• light comes from above
• objects of limited size are usually observed from above (the Ecker cube is often resolved using this observation similar to the example in the beginning of this paper).

There is sometimes an interaction between low level features and high level features. Goldstone [1995] showed that shape categories of objects influenced the perceived colour of an object.

The interpretation also depends on the experience of the viewer. An architect may use his knowledge to interpret drawings in a better way than an ordinary novice.

May [2000] gives some background to high level perception in relation to computer graphics. We should be aware how the viewer understands the concepts in the scene. Walker et al [1994] give an example:the emotion in a facial expression presented at the screen may influence the answers of a questionnaire.

One part of high level visual perception was formulated by Gestalt psychologists more than 100 years ago. They use concepts for visual perception such as proximity, similarity, continuity, closure, figure ground etcetera
The Gestalt laws are usually used for geometrical objects but may also be used for object attributes such as colours with a particular hue, saturation or lightness.

7 Cue theory

Already in the 1700s cues for perception of 3D were developed [Wanger et al. 1992]. Those cues were divided into primary cues, such as convergence and binocular disparity, and into pictorial cues, such as perspective, texture, shading, shadowing and motion. In addition to this, reference pictures (match boxes etc) are also important for determining size and distance.

Visual variables have been discussed by for instance Bertin [1983]. Those variables include things such as shape, size, orientation, intensity, colour and texture. Other visual variables are elevation, shadows, projection, motion, shape, and depth

Combination of features in images is used by humans to interprete the image and extract properties. In vision research this problem is usually referred to as the cue combination problem.
A study made by Wanger et al. [1992] performed three experiments studying the six cues: elevation, object texture, ground texture, shadow, projection and motion. The three experiments were set up for three different tasks, i.e. positioning, rotation and scaling respectively.
It turned out that a substantial positive effect for positioning was given by the cues shadow and perspective. For orientation some positive effect was given by motion and a substantial negative effect was given by perspective projection. For scaling (size) a substantial effect was give by shadow and some effect was given by motion. A negative effect for scaling was given by elevation.

8 Colour

Colour is one of the most important features in a visual presentation.

Perceived size and depth depend on the colour of the object:
• the perceived size of an object is influenced by its colour [Rheingans, Landreth 95]
• the perceived depth of an object is influenced by its colour [Rheingans, Landreth 95]
Experiments show that object appears larger/closer according to:

```
green – blue – yellow – orange – red
```

where red is largest and closest.
Part of the explanation may be due to different refraction of the different wave lengths of the light in the eye. Experimental results also indicates that the effects are diminished by decreasing saturation.

The perceived colour of an object is influenced by the colour of the surroundings referred to as colour interaction, colour induction, or colour assimilation.

Healey et al. [1999] discuss three features important for fast and preattentive detection of targets in a display. The features are colour distance, linear separation and colour category. We discuss them in u*v*-space (designed to be approximately perceptual uniform):
• perceived colour distance, DE* = sqrt((DL*)2 + (Du*)2 + (Dv*)2)
where D is delta (difference) and L* is luminance
• linear separation regards whether the target can be separated from the non-targets by a line in the u*v*-space.
• colour category specifies the categories such as blue, purple and red.

There are two categories of colour coding, nominal coding and ordinal coding.
The nominal coding use colours that are not intended to be ordered in any way. They are only used for identification. Usually a number of five to seven colours are suitable for error free identification.
Ordinal coding use an ordered sequence of colours, which we here call colour scales. Colour hues are not naturally sorted in the same way as lightness and saturation. Nevertheless there are examples of natural colour scales. The hues of the rainbow constitute the spectrum colour scale. A heated radiating body generate subsequent colours as the temperature increase, which constitute the heated-object colour scale (black - red - yellow - white - blue). Other scales are intensity scale and colour saturation scale, where lightness or saturation are varied along the scale.
[Levkowitz and Herman 1992] and [Rogowitz and Treinich 2003] has investigated colour scales.
The rainbow scale may give artifacts such as perceived contours without any discrete transition in the data whereas other transitions may be hard to detect and colours with a high lightness such as yellow may attract attention which is not intended.

There are many possibilities to combine the basic colour scales. Those combinations may be used either to reinforce the same information and hence make it easier for the observer to perceive or to increase the information content of the presentation. Below we give a list of possible combinations. Many of these combinations have been used for a long time in disciplines like geographical maps.
• an elevation map of a region showing heights with shadows can be combined with a usual colour scale showing some non elevation variable.
• double ended colour scales (e.g. based on a scale with intervals of interesting values, uninteresting values, interesting values)
• striped colour scales, discussed in [Rheingans, Landreth 95]
• multiple scales (using several of the parameters above, e.g. both intensity and saturation at the same time), e.g. Levkowitz's optimal colour scale using brightness+hue is linearlized in just noticeable differences [Levkowitz 92].
Further colour scales are discussed in [Rheingans, Landreth 95] and [Ware 98].

9 Texture

Texture is one important feature to visualize a surface. Textures have been studied by researchers in computer graphics, computer vision and cognitive psychology. Sometimes the term pexel is used for perceptual texture to emphasize the perceptual aspects of textures.
Textures can be categorized by several dimensions. Examples of such dimensions are orientation, size, density, contrast, regularity and complexity. Some of these dimensions may be related. For instance we may have
size = constant/density. Density and height are used as separate variables in a study by [Healey et al. 99]. Both variables are detected preattentively.

Ware and Knight [92] discussed the OSC texture space with the purpose of using a three dimensional space for texture properties in similar way as HSV or RGB is used for colour. The three variables for texture would include orientation (O), size (S) and contrast (C). Texture(x,y) is a function with three values:
• orientation (O), the angle of the texture at (x,y) given in the range (0,p)
• size (S), the width of the texture at (x,y). It is the inverse of the density. It can also be described as 1/frequency.
• contrast (C), amplitude of intensity
The authors performed experiments to determine a uniform texture space.

10 Shading

Atherton and Caporael [1985] made a study of judgments by 30 participants. The subjects evaluated spheres rendered with different shading techniques (flat shading, Gouraud shading and Phong shading) and with different number of polygons. The result show just a minor difference between Gouraud shading and Phong shading for the task given to the participants.
Another similar study was set up by Barfield et al. [1988]. The participants were asked to decide whether two objects with different orientations had the same shape.

More advanced computer graphics techniques such as the radiosity methods have been compared in experiments intended to reveal if there is a difference between a radiosity image and a photograph [Meyer et al. 1986]. They used very simple scenes with block like objects. The subjects were unable to distinguish between the radiosity image and the photograph.

New algorithms use the knowledge of perception in order to improve the quality of the images, to prevent calculating things that are invisible and to determine what is most important to calculate if the resources are limited. Prevention of unnessesary calculations may use limitations in frequency, intensity and contrast in the human visual system. A survey is given by McNamara [1999].

11 Levels of realism

Ferwerda [2003] has given three levels of realism in computer graphics:
• physical realism (physical signal is correct)
• photo-realism (physical respons is correct)
• functional realism (the same information is given)

There are more possible levels or categories of realism in computer graphics. A categorization can be made with a point of departure in the different features discussed earlier in this paper. Such categories could include 3D realism, surface (texture) realism, colour realism and illumination realism.

A categorization can also be based on the experience or feelings of the user (viewer). Such experiences could be characterized by the level of e.g. presence, excitement, comfort, recognition and detection.

12 Recommendations and guidelines

Below we give a few examples of advice regarding good pictures in computer graphics and visualization. Guidelines are not always true in the specific case. Sometimes they have to be broken. Guidelines in a limited application as for instance diagram drawing can be more precise while guidelines for images where specific content has not been given is more general. Sources for compiling guidelines include [MacDonald 1999], [Foley et al. 1990] and [Tufte 2001].

Examples of general guidelines for visualization of data:
• have a properly chosen format and design
• use words, numbers, and drawings together
• display an accessible complexity of detail
• often have a narrative quality, a story to tell about the data
• avoid content-free decoration, including chartjunk

Examples of more specific guidelines, here on the use of colours:
• be restrictive in the use of colours. There are many possible mistakes so it might be better to avoid colour if it is not necessary to for the application.
• many people (in the order of 8% of the male population and 0.5% of the female population) have deficient colour perception. Use redundancy for information encoding and presentation.
• equipment will display colours differently dependent of display and printer technology - some kinds of equipment are black and white. Use redundancy for information encoding and presentation.
• the eye and the brain needs intensity differences to distinguish edges. This means that colour hue differences should not be used as the only way of presenting edges. Use intensity differences, saturation differences and/or border lines.
• strong colours, i.e. colours with high saturation, may give afterimages if they are viewed for longer time interval. This means that images with high saturation colours should be avoided for many applications. They can be used to grab attention though.
• perceived hue and intensity are influenced by surrounding coloured areas. This effect can be corrected by adjusting the colours to compensate for this effect (which is hard). It is also possible to use surrounding colours with less saturation.
• colours don't have a natural order from a perceptual point of view. Intensity and saturation have a more natural order. This means that colour scales with only hue differences should be avoided if an ordered relation is presented. Hue differences are better for classification.

13 Conclusions

There are many challenges for computer graphics in the future. Researchers will strive for creating better pictures. An important aspect is to determine if the features created can be seen or are necessary for the viewer. This will require knowledge of perception and related areas.

References

Atherton P R, Caporael L R, A Subjective Judgement Study of Polygon Based Curved Surface Imagery, ACM, CHI '85 Proceedings, April 1985, pp 27–34

Barfield, W, Sandford, J, Foley, J, The mental rotation and perceived realism of computer-generated three-dimensional images, Int J Man-Machine Studies, 29, pp669-684, 1988

Bertin, J, Semiology of Graphics, Univ of Wisconsin Press, Madison, US, 1983

Chalmers, A, Daly, S, McNamara, A, Myszkowski, K, Troscianko, T, Image Qulity Metrics, course notes, course #44, , SIGGRAPH2000

Ferwerda, J A, et al., Psychometrics 101: How to Design, Conduct, and Analyze Perceptual Experiments in Computer Graphics, course notes, course #58, SIGGRAPH2002

Ferwerda, J A, Three varieties of realism in computer graphics. Proceedings SPIE Human Vision and Electronic Imaging'03, 2003

Goldstone, Robert, L, Effects of Categorization on color perception, Psychological Science, vol 6, no 5, sept 1995

Healey, C. G. "Choosing Effective Colours for Data Visualization." In Proceedings IEEE Visualization '96 (San Francisco, California, 1996), pp. 263-270.

Healey, C. G., Booth, K. S., and Enns, J. T. "High-Speed Visual Estimation Using Preattentive Processing." ACM Transactions on Human Computer Interaction 3, 2, (1996), 107-135.

Healey, C. G. and Enns, J. T. "Building Perceptual Textures to Visualize Multidimensional Datasets." In Proceedings IEEE Visualization '98 (Research Triangle Park, North Carolina, 1998), pp. 111-118

Healey C G, Interrante V, Rheingans P, Fundamental Issues of Visual Perception for Effective Perception for Effective Image Generation, course #6, SIGGRAPH'99, August 8-13, 1999

Healey, C. G. and Enns, J. T. "Large Datasets at a Glance: Combining Textures and Colors in Scientific Visualization." IEEE Transactions on Visualization and Computer Graphics 5, 2, (1999), 145-167.

Healey, C. G and Enns, J. T. "Perception and Painting: A Search for Effective, Engaging Visualizations." IEEE Computer Graphics and Applications (Visualization Viewpoints) 22, 2, (2002), 10-15

Horton, W, Top Ten Blunders by Visual Designers, SIGGRAPH Computer Graphics, Nov 1995

Levkowitz H., Herman, G. T., Color scales for image data. IEEE Computer Graphics and Applications, vol 12, no 1 pp72-80, 1992

Mahoney, Diana Phillips, Seeing with the mind's eye, Computer Graphics World, October 2001

McCann, John J., Simultaneous contrast and color constancy: signatures of human image processing, in Davis, S. (ed), Color perception: philosophical, psychological, artistic and computational perspectives, Oxford University Press, 2000

McNamara, A,. STAR: Visual Perception in Realistic Image Synthesis,. In *Eurographics 2000*, Interlaken, Switzerland, August 2000. Eurographics

Meyer, G W, Rushmeier, H E, Cohen, M F, Greenberg, D P, Torrance, K E, An experimental evaluation of computer graphics imagery, Transaction on Graphics, vol 5, no 1, pp30-50, 1986

Mollon, J.D., "Cherries among the Leaves": The evolutionary Origins of Color Vision, in Davis, S. (ed), Color perception: philosophical, psychological, artistic and computational perspectives, Oxford University Press, 2000

May, Jon, Perceptual Principles and Computer Graphics, Computer Graphics Forum, vol 19, no 4, pp 271-279, 2000

Meyer, G.W.; Greenberg, D.P., Color-defective vision and computer graphics displays IEEE Computer Graphics and Applications, vol 8, no 5, pp28-40, 1988

Rheingans Penny, Landreth, Chris, Perceptual Principles for Effective Visualizations, in Perceptual Issues in Visualization, eds G. Grinstein, h. Levkowitz, Springer 1995

Reddy, M, Perceptuallu Optimized 3D Graphics, IEEE Computer Graphics and Applications, vol 21, no 5, 2001

Rogowitz, B, Treinish, L, How NOT to Lie with Visualization, http://opendx.npaci.edu/cds/proceedings96/pravda/truevis.html, visited March 5, 2003

Shacked, R, Kischinski, D, Automatic Lighting Design using a Perceptual Qulity Metric, Eurographics2001, Computer Graphics Forum, vol 20, no 3, 2001

Stilling, J. Die Prüfung des Fabensinnes beim Eisenbaum und Marine-personal, Cassel: Teodor Fischer, 1877

Tufte, E R, The Visual Display of Quantitative Information, Graphics Press, 2001

Wanger, L R, Ferwerda, J A, Greenberg, D P, Perceiving Spatial Relationships in Computer-Generated Images, IEEE Computer Graphics and Applications, vol 12, no 3, pp44-58, 1992

Ware, Colin, Knight, William, Orderable Dimensions of Visual Texture for Data Display: Orientation, Size and Contrast, CHI'92, pp 203-209

Ware, Colin, Knight, William, Using visual texture for information display, ACM Transactions on Graphics (TOG), vol 14 , no 1, 1995

Ware, Colin, Perception and Data Visualization: The Foundations of Experimental Semiotics, Graphics Interface'98, W. Davis, K. Booth, A. Fournier (eds), Canadian Information Processing Society, 1998

Splitting Methods for Dry Frictional Contact Problems in Rigid Multibody Systems: Preliminary Performance Results.

Claude Lacoursière *
VRlab/HPC2N and Computing Science
Umeå Universitet, 901 87 Umeå, Sweden
and
CMLabs Simulations Inc.
420 Notre Dame Street West, suite 505
Montréal, Qc, CANADA, H2Y 1V3

Abstract

A splitting method for solving LCP based models of dry frictional contact problems in rigid multibody systems based on box MLCP solver is presented. Since such methods rely on fast and robust box MLCP solvers, several methods are reviewed and their performance is compared both on random problems and on simulation data. We provide data illustrating the convergence rate of the splitting method which demonstrates that they present a viable alternative to currently available methods.

CR Categories: G.1.6 [Mathematics of Computing]: Optimization—Nonlinear Programming I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

Keywords: physics based modeling, constraints, LCP, dry friction, rigid multibody dynamics

1 Introduction

Rigid multibody dynamics with frictional contacts is a well established topic in interactive graphics and in the engineering literature. There are several papers on the topics each year in the SIGGRAPH proceedings since the late 80s. The application domain in the context of interactive 3D graphics covers operator training for ground vehicles, robotics, gaming, and animation authoring tools, especially for movies.

Several software packages are available on the market but few can simulate rigid multibody systems with dry frictional contacts satisfactorily, though many perform well on problems without dry friction. The issues range from poor stability, lack of robustness in the case of degenerate or ill-posed problems, poor scalability, to anomalous friction forces. Given that a consistent and solvable mathematical model for dry frictional contacts was not published until 1997[Anitescu and Potra 1997], this fact is not very surprising.

Alternative solution methods have been suggested for solving dry frictional contact problems including spring and damper systems, pairwise impulse models[Mirtich and Canny 1995], and many variants based on Linear or Nonlinear Complementarity Problems (LCPs and NLCPs respectively). For the latter, we cite[Al-Fahed et al. 1991][Pang and Trinkle 1996] [Stewart and Trinkle 1996][Anitescu and Potra 1997][Anitescu et al. 1999]. Spring-damper models will not be discussed as they tend to be unstable. Pairwise impulse based models amount to Gauss-Seidel iterative processes and we will provide data on such methods below.

LCP models are split in acceleration based models [Al-Fahed et al. 1991][Trinkle et al. 1997][Pang and Trinkle 1996][Pfeiffer and Glocker 1996][Tzitzouris 2001], and velocity time stepping models [Stewart 1997][Stewart and Trinkle 1996][Anitescu et al. 1999][Anitescu and Potra 1997]. The first type is not guaranteed to be solvable except for very small friction coefficients[Pang and Trinkle 1996]. There are discontinuities inherent in dry friction at transitions from static to kinetic friction as explained in[Stewart and Trinkle 1996]. These lead to impulses i.e., infinite instantaneous accelerations, even when there is no collision. Therefore, a model which requires the computation of accelerations is fundamentally flawed, unless all impulses are detected and processed appropriately. Integrating over the accelerations, it is possible to obtain a solvable velocity time stepping model. This is the basis of the present work.

There is no general solution method which is guaranteed to work for any given LCP, except for total enumeration has complexity of $O(2^n)$ for problems of size n. The most robust method is still that of Lemke[Lemke 1965][Sargent 1978] [Júdice et al. 1992]. Only this method can compute a solution of the solvable dry frictional contact models[Stewart and Trinkle 1996][Anitescu and Potra 1997]. However, Lemke's algorithm only solves these problems after reducing the original mixed linear complementarity problem (MLCP) to an LCP with Schur complements. This is expensive and cannot work on degenerate or ill-conditioned problems. Additionally, Lemke's algorithm cannot reuse a previous solution as an advanced starting point which leads to considerable inefficiency, especially near equilibrium.

Computing dry frictional contact forces is hard because of the coupling between the tangential (friction) and normal components. In the Coulomb model for instance, a contact point is in static friction, with zero tangential contact velocity, until the magnitude of the friction force reaches the product of the friction coefficient with the normal force. Because of this coupling, the problem is no longer a quadratic program (QP) but instead, it is an LCP which does not correspond to a minimization or saddle point problem. However, given the normal contact forces at a given point, finding the tangential friction forces corresponds to solving a QP and alternately, given the tangential friction forces at a contact point, finding the normal contact forces corresponds to solving another QP. This has been used [Šimunović and Saigal 1994][Dostál et al. 2002] to construct operator splitting methods which iteratively estimate the two components. Since box QP solvers can use advanced starting points, it is possible to quickly solve a sequence of QPs which, hopefully, converges to a correct solution of the more accurate LCP model. Operator splitting has already been used for simulating rigid multibody systems; a two pass method was used in[Milenkovic and Schmidl 2001] for instance. Splitting methods are also used in commercial software libraries though the evidence for that is circumstantial. The convergence rate of operator splitting doesn't seem to have been investigated and as we show below, two pass methods do not seem to yield very accurate results and is

*e-mail: claude@hpc2n.umu.se

sensitive on the starting point.

A performance and robustness review of existing box QP solvers has not been published yet, especially with regards to degenerate contacts. In the rigid body case, a single box resting on a plane can lead to a degenerate problem and therefore, a solver which is robust against degeneracy and ill-posed problems is badly needed. The current paper offers such a review.

In what follows, we will concentrate on numerical experiments which demonstrate that good convergence can be obtained in practice, leaving the proof for future work. We will review the performance of solvers for LCPs and QPs with box constraints on both random problems and real data sets extracted from simulations.

2 Linear Complementarity

The LCP is defined for a square, $n \times n$, matrix H and an n dimensional vector q as the problem of finding n dimensional solution vectors z and w such that:

$$0 \le z \perp Hz + q = w \ge 0, \tag{1}$$

where the perpendicularity is understood component wise i.e., $z_i, w_i \ge 0, z_i w_i = 0$. The vector w is sometimes called the "slack" variable. Extensive coverage of this is found in [Murty 1988] and[Cottle et al. 1992].

A simple extension of the LCP is to impose lower and upper bounds on the solution vector z, l and u, respectively, which may be finite or infinite. The residual vector, w, is now split into positive and negative components thus: $w = w_+ - w_-$ with $w_+, w_- \ge 0, w_+ w_- = 0$. This leads to one form of the Mixed LCP which we called the box LCP. The definition is as follows:

$$\begin{aligned} Hz + q &= w_+ - w_-, \\ 0 \le z - l \perp w_+ \ge 0, &\quad 0 \le u - z \perp w_- \ge 0. \end{aligned} \tag{2}$$

A review of many algorithms for solving LCPs is found in [Júdice 1994].

3 Multibody Dynamics

We consider a multibody system with generalized coordinates $q \in \mathbb{R}^n$, generalized velocities $v = \dot{q}$, and mass matrix $M(q) \in \mathbb{R}^{n^2}$. The generalized forces acting on the system are $F \in \mathbb{R}^n$ (including the non-inertial forces). The system is subject to holonomic constraints, $\Phi(q,t) = 0$, and inequality constraints, $\Xi(q,t) \ge 0$. The Jacobians for these are written as G and N, respectively, and the Lagrange multiplier associated with them are written λ and ν, resp. Velocity constraints are not covered to simplify the notation. Detailed description of constrained dynamical systems are found in [Goldstein 1980][Layton 1998]. A more detailed description of the notation used here for multibody systems is found in [Anitescu and Potra 1997]. We are assuming descriptor form here but mixed representations are possible[Trinkle et al. 1997].

Using this notation, we have the following equations of motion for a multibody system with frictionless contacts:

$$\begin{aligned} M\dot{v} - G^t\lambda - N^t\nu - F &= 0 \\ \Phi(q,t) &= 0 \\ 0 \le \nu \perp \Xi(q,t) &\ge 0. \end{aligned} \tag{3}$$

This system in Eq. (3) is a form of differential algebraic equation (DAE) (see ref. [Hairer and Wanner 1996] for instance). However, due to the inequality constraints, this is not a standard problem. In[Stewart and Trinkle 1996], it is shown that these systems are differential inclusions. We shall concentrate on the solution methods for the static problem, common to all numerical integration procedures.

Numerical integration of Eq. 3 requires solving the following linear system:

$$\begin{bmatrix} M & -G^t & -N^t \\ G & 0 & 0 \\ N & 0 & 0 \end{bmatrix} \begin{bmatrix} v \\ \lambda \\ \nu \end{bmatrix} + \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \delta \end{bmatrix}, \quad 0 \le \nu \perp \delta \ge 0. \tag{4}$$

The vectors a, b, c depend on the discretization details. The system of equations in Eq. 4 form a mixed linear complementarity model (MLCP).

4 Dry Frictional Contact Models

For a pair of bodies with indicies i_1, i_2 which are in close proximity, it is assumed there is at least one signed distance functions: $\xi^{(j)}(q^{(i_1)}, q^{(i_2)})$ such that $\xi^{(j)} > 0$ if the bodies are separated, $\xi^{(j)} = 0$ if they are touching and $\xi^{(j)} < 0$ if they are penetrating. This function $\xi^{(j)}$ represents a potential point of contact. In practice, several of these signed distance functions are used for a given pair of rigid bodies. To compute the normal force required to prevent interpenetration, we impose the kinematic constraint $\xi^{(j)} \ge 0$ with a Lagrange multiplier $\nu^{(j)} \ge 0$. Contacts are non adhesive and therefore, $\nu^{(j)} \ge 0$ and is complementary to $\xi^{(j)} \ge 0$.

The gradient of $\xi^{(j)}$ defines a vector $\bar{n}^{(j)}$ normal to the tangent contact plane. This is spanned by a set of basis vectors $\{d^{(j,1)}, d^{(j,2)}, \ldots, d^{(j,n_d)}\}$. The simplest case, we use just two orthogonal vectors but some models require the use of a large number of non-orthogonal vectors to reduce anisotropy[Anitescu and Potra 1997]. Using these, we can compute the projection $D^{(j)}$ such that the relative tangential velocity in the contact plane is given by $\bar{v}^{(j)} = D^{(j)}v$ and the tangential contact force is given by $D^{(j)T}\beta^{(j)}$.

Static friction corresponds to zero contact velocity i.e., $D^{(j)}v = 0$. For kinematic friction, the magnitude of the tangential contact force is given by the product of the kinetic friction coefficient and the magnitude of the normal force at the contact point, $\|\beta^{(j)}\| = \mu_k^{(j)}\nu^{(j)}$, and the direction is anti-parallel to the tangential contact velocity. We only cover the case of isotropic friction here and we use only one friction coefficient for both cases. Two simplifications are possible.

First, one can linearize the norm operator. A vector in the tangent plane can be written as: $x^{(j)} = \sum_k \alpha_k d^{(j,k)}$ with $\alpha_k \ge 0$, with only few non-zero α_k. We introduce $E^{(i)} = (1, 1, \ldots, 1)^T$) such that $\|x^{(j)}\| \simeq {E^{(j)}}^T x^{(j)} = \sum_k \alpha_k$. A linearized approximation of the Coulomb friction model consists of the following set of complementarity conditions:

$$\begin{aligned} 0 \le D^{(j)}v + E^{(j)}\sigma^{(j)} \perp \beta^{(j)} &\ge 0 \\ 0 \le \mu^{(j)}\nu^{(j)} - E^{(j)T}\beta^{(j)} \perp \sigma^{(j)} &\ge 0. \end{aligned} \tag{5}$$

The second equation tells us that as long as the magnitude of the friction force $|\beta^{(j)}|$ is less than the product of the friction coefficient and the normal force, the sliding velocity $\sigma^{(j)}$ vanishes and the first equation imposes that constraint as $D^{(j)}v = 0$. When the magnitude of the tangential force reaches the maximum, we get a non-zero sliding velocity and the first block of complementarity conditions picks a direction for the tangential force which is nearly anti-parallel to the sliding.

This model[Stewart and Trinkle 1996][Anitescu and Potra

1997], leads to the following MLCP:

$$\begin{bmatrix} M & -G^T & -N^T & -D^T & 0 \\ G & 0 & 0 & 0 & 0 \\ N & 0 & 0 & 0 & 0 \\ D & 0 & 0 & 0 & E \\ 0 & 0 & U & -E^T & 0 \end{bmatrix} \begin{bmatrix} v \\ \lambda \\ \nu \\ \beta \\ \sigma \end{bmatrix} + \begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \rho \\ \delta \\ \eta \end{bmatrix}, \quad (6)$$
$$0 \leq \begin{bmatrix} v^t & \beta^t & \sigma^t \end{bmatrix}^t \perp \begin{bmatrix} \rho^t & \delta^t & \eta^t \end{bmatrix}^t \geq 0,$$

where a,b,c,d,e depend on the discretization of choice, the matrix U is the agglomeration of the friction coefficients, $U = \text{diag}(\mu^{(1)}, \mu^{(2)}, \ldots \mu^{(n_c)})$ for n_c contact points, and E is the agglomeration of the linearized norm operator: $E = \text{diag}(E^{(1)}, E^{(2)}, \ldots E^{(n_c)})$.

This MLCP can be solved using Lemke's method but in order to do this, we need to take two Schur complements. Details for this are found in [Anitescu et al. 1999]. Computing this matrix is far from trivial in terms of numerical work. However, though the Lemke algorithm can process the reduced problem, no extension is guaranteed to solve the original MLCP of Eq. (6).

The second option is to impose a fixed bound on the tangential forces based on an estimate of the normal force. Using two perpendicular directions, $d^{(j,1)}, d^{(j,2)}$, we impose the box bounds $-\mu^{(j)}\bar{v}^{(j)} \leq \beta^{(j,i)} \leq \mu^{(j)}\bar{v}^{(j)}$ for $i = 1,2$, where $\bar{v}^{(j)}$ is an approximation of the expected normal force for the given contact point. The constraint equations expressing this model are as follows:

$$\begin{aligned} Dv - \bar{v}_+ + \bar{v}_- &= 0, \\ 0 \leq \beta - \underline{\beta} \perp \bar{v}_+ \geq 0, \quad 0 \leq \overline{\beta} - \beta \perp \bar{v}_- &\geq 0 \end{aligned} \quad (7)$$

where $\overline{\beta}, \underline{\beta}$ are the lower and upper bounds of β respectively, and $\bar{v}_+$ and $\bar{v}_-$ are the positive and negative components of the tangential contact velocity respectively. This model is dissipative and it has most of the properties of the Coulomb model. However, it exhibits anisotropy and has the wrong transition point if the estimate is too far. This can be turned into an iterative scheme though, which can converge to the correct answer. This is a form of operator splitting which requires fast solution of box constrained MCLPs and we provide performance data on this below.

The box friction model leads to the MLCP:

$$\begin{bmatrix} M & -G^T & -N^T & -D^T \\ G & 0 & 0 & 0 \\ N & 0 & 0 & 0 \\ D & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v \\ \lambda \\ \nu \\ \beta \end{bmatrix} + \begin{bmatrix} \tilde{a} \\ \tilde{b} \\ \tilde{c} \\ \tilde{d} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \tilde{\rho} \\ \tilde{\delta} \end{bmatrix}, \quad (8)$$
$$0 \leq \nu \perp \tilde{\rho} \geq 0, \quad 0 \leq \beta - \underline{\beta} \perp \tilde{\delta} \geq 0, \quad 0 \leq \overline{\beta} - \beta \perp \tilde{\delta} \leq 0.$$

If the estimates for $\underline{\beta}$ and $\overline{\beta}$ are accurate and if the directions $d^{(j,k)}$ are well-chosen, a solution of the box MLCP will also solve the MLCP of Eq. (6).

5 Solvers for Complementarity Problems

There are three types of LCP solvers: pivoting methods, Newton methods, and iterative methods. An extensive review of these algorithms is available in[Júdice 1994]. Except for special cases, LCPs are NP hard problems. A statistical study of solver performance is therefore appropriate.

The pivoting methods include Lemke's almost complementary pivot method (see [Murty 1988], ch. 2), Cottle-Datzig's principal pivot method[Cottle 1968], Keller's principal pivot method[Keller 1973], and Murty's principal pivot method[Murty 1974] among many others. The amount of work done per iteration amounts to a Gauss-Jordan pivot operation[Golub and Van Loan 1996] which is of $O(n^2)$ where n is the size of the problem.

Lemke's method solves the largest class of problems namely, those defined with copositive plus matrices i.e., matrices H for which $y \in \mathbb{R}^n, y \geq 0 \leftarrow Hy \geq 0$ and such that for $y \geq \in \mathbb{R}^n$, if $y^T My = 0$ then, $(H + H^T)y = 0$. This class includes positive semidefinite matrices. Keller's and Cottle-Dantzig's principal pivot methods are guaranteed to work on P_0 matrices which are those for which all principal minors are non-negative. This class also includes positive semi-definite matrices. Murty's method only works on P matrices, those for which all principal minors are positive which includes positive definite matrices. With the exception of Murty's principal pivot methods, none of the pivot methods can be started at an advanced point. As we show below, both Lemke's and Keller's method exhibit good performance, performing roughly n pivot operations on average, at least on the class of problems we tested. Murty's and Cottle-Dantzig's methods seem to perform erratically, executing many times more than n pivot operations on problems of size n. All these methods can be extended to cover MLCP with box constraints.

Newton's method can be applied to MLCPs[Kelley 1995] and in particular, we have used[Zhang and Gao 2003][Li and Fukushima 2000]. A block pivot method[Kostreva 1978][Júdice and Pires 1994] can be shown to be equivalent to a Newton method without smoothing and without line search. This has been used extensively. The methods with smoothing and line search are more recent and haven't been used extensively yet but are presumed to be more robust but are more complicated to tune with roughly a dozen free parameters. Newton methods require solving a linear system of size n at each step and therefore, each iteration has approximate cost $O(n^3)$. All Newton-type methods can start from an advanced point i.e., a point which is hoped to be near the solution. All Newton methods can solve P_0 problems but work better on P problems.

For iterative methods, the matrix H is decomposed as $H = D + L + U$ where D is a block diagonal matrix, L is a strictly lower triangular matrix and U is a strictly upper triangular matrix. We then solve LCPs corresponding to one block of equations, keeping the other ones fixed:

$$0 \leq D_{jj} z_j^{(k+1)} + q_j + L_{jj'} z_{j'}^{(k+1)} + U_{jj'} z_{j'}^{(k)} \perp z_j^{(k+1)} \geq 0. \quad (9)$$

Here, j is the set of indicies corresponding to a block and j' is the set of all other indicies. This is a block Gauss-Seidel scheme.

The Gauss-Seidel method is very attractive for its simplicity. However, the convergence rate is ρ^m where ρ is the condition number of the matrix $D^{-1}(L+U)$. Though $\rho < 1$ when H positive definite, it is often very near 1. When working on random problems, condition numbers exceeding $1E7$ made the method unusably slow. On simulation data extracted from a piling problem, the method stagnated for thousands of iterations.

All pairwise methods such as[Mirtich and Canny 1995] and the many variants thereof e.g., [Guendelman et al. 2003], are essentially Gauss Seidel processes and are expected to suffer from low accuracy, especially when they are limited to one or two sweeps through the constraints.

6 Implementation Details

The principal pivot methods of Keller and Lemke, as well as the smoothed Newton method were implemented in Octave. The other methods were implemented in C/C++, linking to BLAS, LAPACK, and GSL, the GNU Scientific Library. Wrappers where then written in C++ to link these in Octave.

The framework used for simulating rigid multibody systems was the Vortex library from CMLabs Simulations (see `http://www.cm-labs.com`).

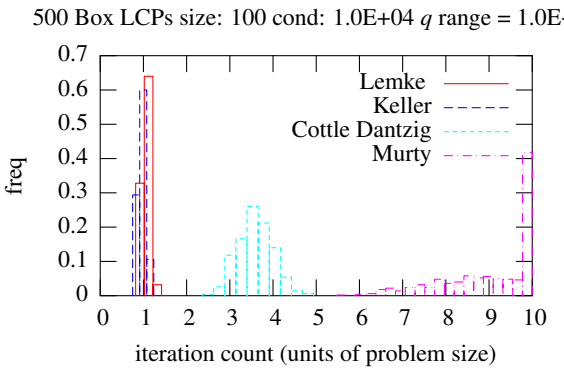


Figure 1: Histogram data for pivot methods on box LCP. Iteration count is in units of the problem size.

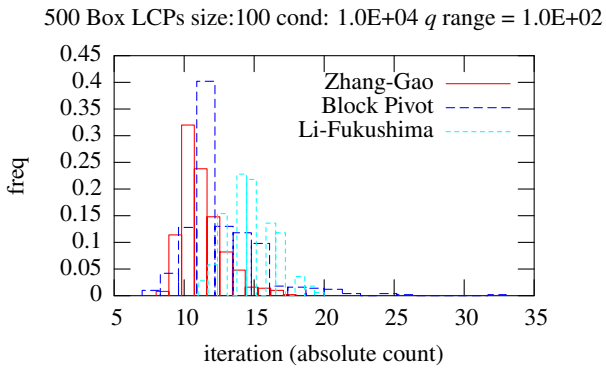


Figure 2: Histogram data for two Newton methods on box LCP of size 100. Iteration count is in absolute units.

We did not perform timing analysis for this report but concentrated on overall iteration count. This provides worse case analysis to select a viable method which will eventually be optimized.

7 Experimental Results

We tested box LCP solvers on a set of random problems generated using the method described in[Alefeld et al. 1999]. All matrices have full rank in these tests. For the few methods which can be started at an advanced point near the solution, we concentrated on the worse case scenario, starting with a z vector at the lower bounds.

Data is presented using histogram because no LCP solver is guaranteed to process a given problem in less than 2^n operations. Histograms give an idea of the expected operation count for a given family of problems. Sharp peaks indicate that the given solver is nearly deterministic in terms of operation count. Broad distribution indicate that a given solver can take wildly different amounts of computation time to process different problems. These graphs represent the frequency of problems solved against iteration count. For pivot method, we used relative iteration counts, normalized to problem size. For Newton methods, the iteration counts are nearly independent of problem size and we use absolute iteration counts.

Data for random box LCPs of size 100 is shown in Fig. 1. We found sharp peaks for both Lemke's and Keller's method near n pivot steps. The Cottle-Dantzig method shows a broad peak near $2n$ pivot steps and Murty's method is slightly worse. The sharp peaks remain for Lemke's and Keller's method for larger systems but the distribution for Murty's and the Cottle Dantzig methods stretch out further to higher iteration counts and become much broader as well. This suggests that Lemke's and Keller's method are near deterministic in performance. For bigger problems, Murty's and Cottle-Dantzig's methods are not usable. Murty's method can be started at an advanced point near the solution in which case it might offer better performance but the worse case scenario is not promising.

For Newton-type methods, we present two solvers: a block principal pivot method [Kostreva 1978] and a globally convergent smoothed Newton method[Zhang and Gao 2003]. For these the iteration count is expected to be independent of problem size and the distribution should be sharp. Results for box LCPs of size 100 are show in Fig. 2. The block pivot distribution is slightly skewed towards higher iteration count because it can actually cycle. Our implementation includes cycle detection and restart as described in[Júdice and Pires 1994].

Next we turn to the iterative block Gauss Seidel solver. On random problems, this behaved just as per the theory so we omit the

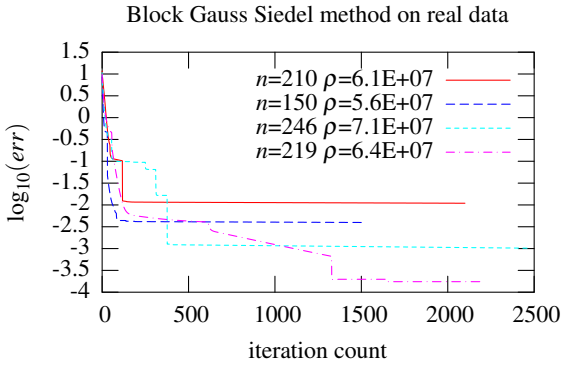


Figure 3: Residual decay as a function of iteration for a stacking problem with 40 cylinders. Each iteration is a one complete sweep over the diagonal elements.

results. Essentially, for condition numbers greater than 1E7, this is not usable. Scaling both the rows and the columns did not improve the results much. Surprisingly, when this is used to simulate stacks of rigid bodies, the anomalies are not immediately apparent despite large residual norms. This is appealing since one can quit early after a fixed number of iteration. Nevertheless, the convergence rate is too low for this to be a good alternative and the method often stagnates as is seen in Fig 3.

We used the Vortex Toolkit from CMLabs and modified the engine to use the Keller method. Using both box friction and scaled box friction model, we simulated stacking problems with 40 identical cylindrical logs falling on each other. This leads to degenerate systems. The prototype solvers were then tested on data extracted from the simulations. A still frame from the simulation is shown in Fig. 4. Results a are shown below in Fig. 5 for pivot methods and Fig. 6 for Newton methods.

Finally, we tested the convergence of the operator splitting scheme. First, for the case of a box sliding down an inclined plane, we found that a two pass scheme starting with zero friction yields the correct Coulomb relation. However, a two pass scheme starting with infinite friction doesn't behave correctly: it needs about 5 iterations to converge to the correct answer. This demonstrates that in general, a multi-pass method is necessary. Results are summarized in Fig. 7.

On a more complicated problem with a stack of 40 cylinders, we

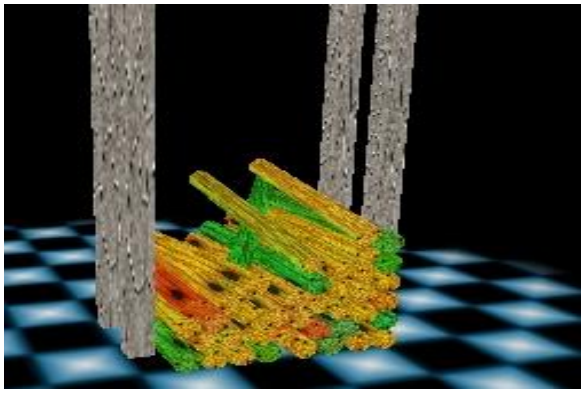


Figure 4: A stack of 40 indentical cylindrical logs falling under gravity.

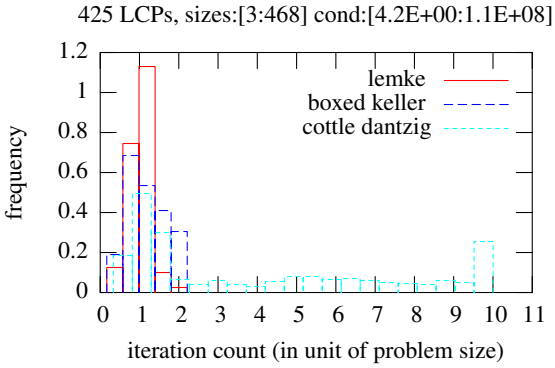


Figure 5: Histogram data for three pivot methods on box LCP problems extracted from a simulation with 40 cylinders forming a heap. The first two peaks correspond to Lemke's method which never seems to take more than n iterations for these problems. Keller's method takes at most $1.2n$ iterations but Cottle-Dantzig's principal pivot method has a long tail up to $4n$ iterations.

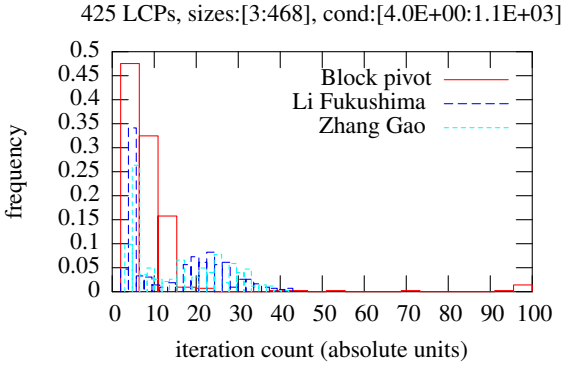


Figure 6: Histogram data for three Newton methods on box LCP problems extracted from a simulation with 40 cylinders forming a heap. A diagonal perturbation of 1E-3 was applied on the diagonal to regularize the problems.

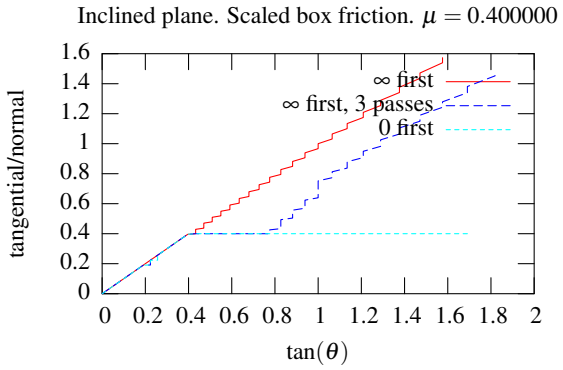


Figure 7: Inclined plane experiment with scaled box friction, using either frictionless or infinitely sticky first pass. The y axis is the ratio of tangential to normal forces and the x axis is the tangent of the angle of the inclined plane.

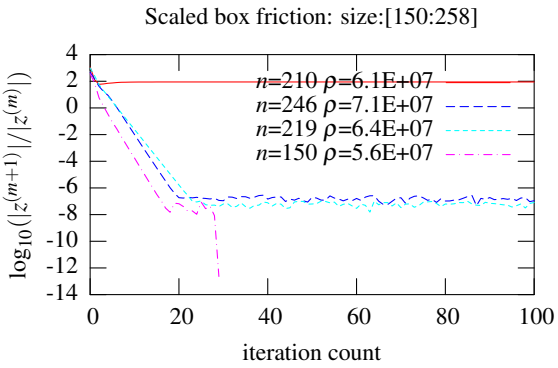


Figure 8: Convergence data for the operator splitting scheme using a smoothed Newton method for each problem. The error norm is the difference between the iterates.

found the splitting scheme could converge to a fixed point within less than 10 iterations as shown on Fig. 8, but not in all cases. The convergence rate of the splitting scheme is linear which is not the best news. Also, there appears to be stagnation after an error of 10E-8 is reached. This should be investigated further.

8 Conclusion and Future Work

We have shown that an operator splitting method based on box MLCP solvers is a viable alternative for solving dry frictional contact problems in rigid multibody systems. For this to work, we need a fast and robust solver for box MLCPs. Some classical pivot methods are robust and relatively efficient but not appropriate to solve really large systems as they cannot use advanced starting points or iterative techniques. Newton methods appear to be robust but more work is required to make them efficient. In particular, more research is needed to speed up the computation of the search direction. The performance results also indicate that popular solvers such as the Cottle-Dantzig principal pivot method and the block Gauss Seidel iterative method adapted for solving MLCP are far from optimal choices when it comes to efficiency or accuracy.

The relevance to interactive graphics is that a robust method

which is guaranteed to work with a more or less guaranteed bound on the amount of work to update a system with a given number of bodies and constraints is essential for the context of real-time interactive applications. We have identified solvers for MLCP with box constraints which meet these requirements though at this time, the expected complexity is still too high at $O(n^3)$, where n is the total number of constraints.

More work is also needed to improve the convergence rate of the splitting method. It might be possible to include extra equations in a smoothed Newton scheme to make the convergence rate quadratic instead of linear as is reported here. Box friction models are notoriously anisotropic, a problem which was not presented above but which we intend to address as well.

9 Acknowledgments

This work was supported by the "Objective 1 Norra Norrlands" EU grant awarded to VRlab. This research was conducted using the resources of High Performance Computing Center North (HPC2N). Deep thanks to Michael M. Kostreva who guided me through the LCP literature and showed much interest in this work, providing for many stimulating discussions. Further thanks go to Kenneth Holmlund and Bo Kågström who gave me an opportunity to start the work reported here, and to my colleagues at CMLabs Simulations who supported my move back to university.

References

AL-FAHED, A. M., STRAVROULAKIS, G., AND PANAGIOTOPOULOS, P. 1991. Hard and soft fingered robot grippers. the linear complementarity approach. *Z. angew. Math. u. Mech. 71*, 7/8, 257–265.

ALEFELD, G. E., CHEN, X., AND POTRA, F. A. 1999. Numerical validation of solutions of linear complementarity problems. *Numerische Mathematik 83*, 1, 1–24. Also available as preprint as report-102.ps from ftp.math.uiowa.edu.

ANITESCU, M., AND POTRA, F. 1997. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics 14*, 231–247.

ANITESCU, M., POTRA, F. A., AND STEWART, D. E. 1999. Time-stepping for three-dimensional rigid body dynamics. *Computer Methods in Applied Mechanics and Engineering 177*, 183–197.

COTTLE, R. W., PANG, J.-S., AND STONE, R. E. 1992. *The Linear Complementarity Problem*. Computer Science and Scientific Computing. Academic Press, New York.

COTTLE, R. 1968. The principal pivot method of quadratic programming. In *Mathematics of the Decision Sciences part I.*, American Mathematical Society, Providence, RI., G. Dantzig and A. V. Jr., Eds., vol. 11-12 of *Lectures in Applied Mathematics*, American Math. Society, 144–162.

DOSTÁL, Z., HASLINGER, J., AND KUČERA, R. 2002. Implementation of the fixed point method in contact problems with coulomb friction based on a dual splitting type technique. *J. of Comp. and Appl. Maht. 140*, 245–256.

GOLDSTEIN, H. 1980. *Classical Mechanics*, second ed. Addison-Wesley, Reading, MA, USA.

GOLUB, G. H., AND VAN LOAN, C. F. 1996. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins Press, Baltimore.

GUENDELMAN, E., BRIDSON, R., AND RONALD FEDKIW. 2003. Nonconvex rigid bodies with stacking. In *Proceedings of the ACM SIGGRAPH 2003*, ACM Transactions on Graphics, J. Hart, Ed., vol. 22, 871–878.

HAIRER, E., AND WANNER, G. 1996. *Solving Ordinary Differential Equations II: Stiff and Differential Algebraic Problems*, second revised edition ed., vol. 14 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, Heidelberg, New York, London, Paris, Tokyo, Hong Kong.

JÚDICE, J. J., AND PIRES, F. M. 1994. A block principal pivoting algorithm for large-scale strictly monotone linear complementarity problems. *Computers Ops Res. 21*, 5, 587–596.

JÚDICE, J. J., MACHADO, J., AND FAUSTINO, A. 1992. An extension of the lemke's method for the solution of a generalized linear complementarity problem. In *System Modelling and Optimization*, Springer-Verlag, P. Kall, Ed., vol. 180 of *Lecture Notes in Control and Information Sciences*, 221–230. Proc., 15th IFIP TC7 Conference, Zurich, September 1991.

JÚDICE, J. J. 1994. Algorithms for linear complementarity problems. In *Algorithms for Continuous Optimization*, Kluwer Academic Publishers, E. Spedicato, Ed., vol. 434 of *NATO ASI Series C, Mathematical and Physical Sciences, Advanced Study Institute*, NATO, 435–475.

KELLER, E. 1973. The general quadratic optimization problem. *Mathematical Programming 5*, 311–337.

KELLEY, C. T. 1995. *Iterative Methods for Linear and Nonlinear Equations*, vol. 16 of *SIAM Frontiers*. SIAM Publ., Philadelphia.

KOSTREVA, M. M. 1978. Block pivot methods for solving the complementarity problem. *Lin. Alg. Appl.*, 21, 207–215.

LAYTON, R. A. 1998. *Principles of Analytical System Dynamics*. Mechanical Engineering Series. Springer-Verlag, Berlin, Heidelberg, New York, London, Paris, Tokyo, Hong Kong.

LEMKE, C. E. 1965. Bimatrix equilibrium points and mathematical programming. *Management Science 11*, 681–689.

LI, D., AND FUKUSHIMA, M. 2000. Smoothing newton and quasi-newton methods for mixed complementarity problems. *Comp. Opt. and Appl. 17*, 203–230.

MILENKOVIC, V. J., AND SCHMIDL, H. 2001. Optimization-based animation. In *SIGGRAPH 2001 Conference Proceedings, August 12–17, 2001, Los Angeles, CA*, Computer Graphics Proceedings, Annual Conference Series, 37–46.

MIRTICH, B., AND CANNY, J. 1995. Impulse based simulation of rigid bodies. In *Symposium on Interactive 3D Graphics*, ACM Press, New York, ACM, 181–188.

MURTY, K. G. 1974. Note on a bard-type scheme for solving the complementarity problems. *Opsearch 11*, 123–130.

MURTY, K. G. 1988. *Linear Complementarity, Linear and Nonlinear Programming*. Helderman-Verlag, Heidelberg.

PANG, J.-S., AND TRINKLE, J. C. 1996. Complementarity formulations and existence of solutions of dynamics multi-rigid-body contact problems with coulomb friction. *Journal of Mathematical Computing 73*, 2, 199.

PFEIFFER, F., AND GLOCKER, C. 1996. *Multibody Dynamics with Unilateral Contacts*. Wiley Series in Nonlinear Science. John Wiley & Sons, New York, London, Sydney.

SARGENT, R. W. H. 1978. An effcient implementation of the lemke algorithm and its extension to deal with upper and lower bounds. *Mathematical Programming Study 7*, 36–54.

STEWART, D. E., AND TRINKLE, J. 1996. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal for Numerical Methods in Engineering 39*, 2673–2691.

STEWART, D. E. 1997. Existence of solutions to rigid body dynamics and the painlevé paradoxes. *C. R. Acad. Sci. Paris 325, Série I*, 689–693.

TRINKLE, J. C., PANG, J., SUDARSKY, S., AND LO, G. 1997. On dynamic multi-rigid-body contact problems with coulomb friction. *Z. angew. Math. u. Mech. 77*, 4, 267–279.

TZITZOURIS, J. A. 2001. *Numerical Resolution of Frictional Multi-Rigid-Body Systems via Fully Immplicit Time-Stepping and Nonlinear Complementarity*. PhD thesis, Johns Hopkins University.

ŠIMUNOVIĆ, S., AND SAIGAL, S. 1994. Frictional contact formulation using quadratic programming. *Computational Mechanics 15*, 173–187.

ZHANG, L., AND GAO, Z. 2003. Quadratic one-step smoothing newton method for p_0 lcp without strict complementarity. *Appl. Math. and Comp. 140*, 367–379.

3D Visualization and 3D and Voice Interaction in Air Traffic Management

Marcus Lange, Jonas Hjalmarsson, Matthew Cooper*, Anders Ynnerman.
Norrköping Visualization and Interaction Studio, Linköpings Universitet, Norrköping, Sweden.
Vu Duong
Experimental Research Centre, Eurocontrol, Brétigny-sûr-Orge, France

Abstract

This paper describes the implementation of a 3D VR system for real time visual representation and manipulation of data in air traffic management and control. The system has been designed and implemented in collaboration with experts in the Eurocontrol Centre for research and development in air traffic control and is currently undergoing evaluation by air traffic control staff.

The system will enable us to determine what benefit, in terms of enhanced understanding and clarity of perception, such 3D displays, combined with enhanced information presentation, can provide to the controller. It is hoped that improvements in this area will permit more efficient and safe management of more aircraft over a wider airspace. Interaction schemes have initially been centred around the use of 3D interactors such as a 3D pointer but we have recently implemented a scheme for voice recognition which allows the user access to a much wider range of commands and actions without recourse to a static keyboard or complex button combinations. The exploitation of carefully selected voice controls frees the user to work within the VR environment using little or no hand-based interaction.

We intend to continue development of this system and expect that it will form an evaluation test bed for a wide range of new VR, 3D and other interaction technologies within this application area in the future.

CR Categories: I3.7 [Three-Dimensional Graphics and Realism]: Virtual Reality, J7 [Computer in Other Systems]: Command and control,

Keywords: Virtual Reality, Air Traffic Control, Voice Recognition.

*e-mail:matco@itn.liu.se

1. INTRODUCTION

The management of air traffic across the wide areas used in international air traffic routes is a large and growing problem with the ever increasing numbers of passengers and flights in operation. The systems used for air traffic management, however, are only adopting new technologies quite slowly with the modern controller being faced with a remarkably similar working environment to that which has been in use for more than 30 years. Despite the fact that the management of air traffic is obviously a three dimensional problem and despite long discussions[1][5] of what benefits might be gained from its use, to date no use of 3D displays has been made in production tools for commercial air traffic management. This results in a system where the controller, being provided with only a 2D representation of the data, must construct and hold the 3D model in their head: a difficult task with a large number of active aircraft spread across the wide airspace which they must manage.

This paper describes an ongoing collaborative development between the Norrköping Visualization and Interaction Studio and the Innovative Research group at the Eurocontrol research centre. The projects seeks to explore the application and benefits of 3D (stereo) VR display and interaction technologies with a view to determining the qualities required to produce an effective 3D information visualization environment for the air traffic controller. The currently implemented system, targeted at semi-immersive displays such as workbench displays and VR theatres, makes use of VR techniques to provide an environment within which an air traffic controller can observe and monitor a large number of aircraft over a wide area, being kept aware of the many complex factors about their planned routes which may affect the future planning of the flight paths, and can use 3D interaction methods to select and re-route aircraft interactively as the data are updated in real time.

The recent addition of voice control to the system has opened up the possibility of reducing the number of commands which must be mapped into the 3D interaction devices, the number of which is so high after two years of development that the user must frequently make use of a keyboard to be able to control the whole system. The voice control is now capable of controlling all features of the system with the exception of some of the more complex navigational tasks and for selecting aircraft and performing routing operations on them. For these tasks a 3D wand remains the interaction method of choice.

2. DESIGN AND IMPLEMENTATION

To provide an effective test-bed for 3D VR interaction and visualization in air traffic management we have developed a system which can provide a controller with a 3D environment showing all of the aircraft active in the controller's particular region of interest and those whose routes will take them through it during their flight time. The system is based around Stockholm's Arlanda airport which is a medium capacity airport supporting regional, national and some international air traffic. The flight information which we have obtained provides us with a complete set of inbound and outbound flights for Arlanda across approximately 60 minutes of a weekday morning and includes a range of different aircraft types and flight distances.

2.1 The visualization problem

Aside from the problem of the number of aircraft which are active in the airspace, each flight present in the display is affected by a large number of associated factors which control the behaviour of the aircraft and define the way in which the controller will manage the flight. This information includes such factors as the aircraft type, its current airspeed and altitude, the extent of its vortex wake and how it will be affected by those of the other aircraft preceding it, its take off and expected landing times, the name of the airline who operate this aircraft and its flight number. External information such as weather data, both forecast and reported, can also be extremely important when controllers wish to make routing decisions for the aircraft under their control.

In the existing air traffic control scenario, with which we are all familiar from films and documentaries, the air traffic controller is presented with a limited amount of information through a text block attached to glyph 'blips' on a 2D radar display. Thus the sum of information to which the controller has immediate access will typically be the flight number, position, direction, speed and altitude. All other information is likely to be provided through another source, either a separate computer display or even on paper.

Within our immersive VR environment we wish to present all of this information within the scene with no recourse required to external information sources. To force the user to switch from the immersive display to a separate information source would both be time consuming, taking their attention away from the active flights over which they are watching, and would damage the sense of presence which such VR systems as this can engender and from which we hope to benefit through an enhanced sense of awareness of the 3D scene. Thus all of the required information must be attached to the aircraft in a manner which makes it clear and easy for the controller to interpret with a minimum of searching required.

2.2 Visual information representations

The 3D environment that we have developed represents the aircraft as easily recognizable 3D geometries for the different aircraft types and we have attempted to map airline liveries onto the models, where possible, to provide information about the airline. Flight paths are indicated by colour-coded tubes connecting waypoint 'posts' which indicate the real and virtual waypoints which the aircraft are instructed to fly past or through. The waypoints also have an associated altitude defined as a flight level (1000 foot intervals). This three dimensional information, using models which have been scaled up enormously from their real-world size, provides useful depth cueing information to the controller and makes the aircraft models (and hence their types) visible when the controller is looking at the scene in a global overview.

To assist in conflict avoidance and resolution the controller is provided with the facility to display information about each aircraft's position at a future time. The future position, at one minute intervals, is displayed using a colour-coded tube which precedes the aircraft as it moves and indicates its planned route through branches and joins in the trajectory network.

Height information regarding the planned altitude for each flight at each waypoint is included using colour-coding on the waypoint posts and using a translucent 'curtain' below the flight paths to make the individual flight levels visible to the user. We have also experimented with displaying the actual current altitude of each flight using a colour component attached to some element of the 3D structure of the aircraft, typically the wings and, perhaps, the tailplane but the number of flight levels and hence the number of distinct colours required makes it difficult to make this information sufficiently apparent. At present some additional information is attached to the aircraft as a text tag which conveys the flight number and current altitude but it is intended to explore other ways to include this information in the future.

A close up view of the active system, showing all of the features described above is shown in Figure 1.

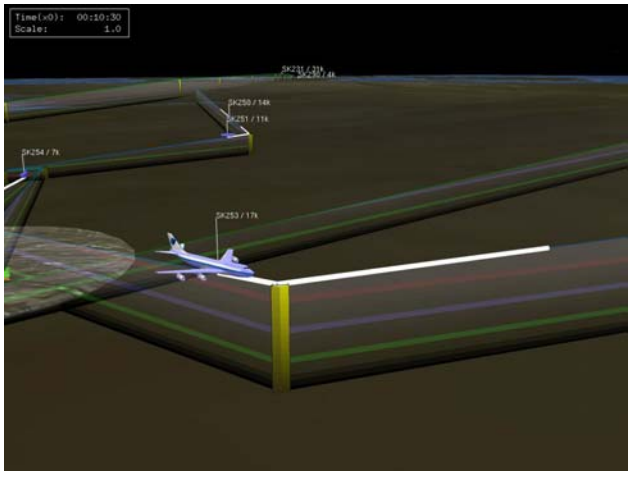


Figure 1: Close up view showing aircraft models, flight path information and estimated future flight positions.

2.3 Navigation and orientation

To provide usable navigation and orientation information for the controller we have included a limited amount of geographical information in the scene. The local terrain is represented as a three dimensional surface based on a 1km resolution map obtained from the US Geographical Information Survey. No attempt to make the map photorealistic has been made since this would provide useful information to the controller. Instead a simple colour map based on height has been included to highlight major orientation features such as water and substantial hills and mountains. The region of the terrain around Arlanda is quite flat so the map typically shows up as a relatively uniform green colour but the coastline is distinctive and provides excellent orientation information for the controller as they move their viewpoint across the scene.

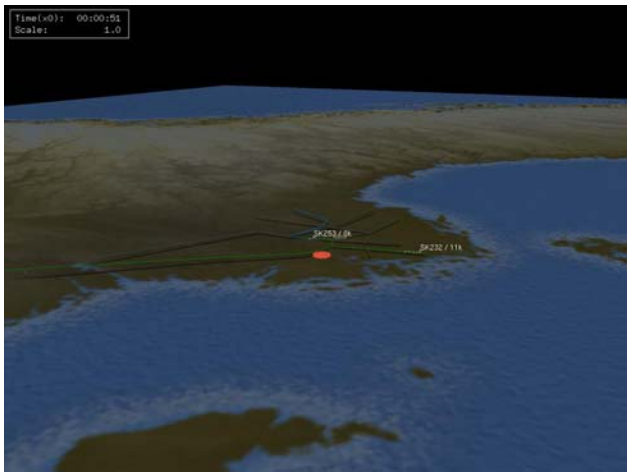


Figure 2: 3D geography over Scandinavia, coloured by height

Terrain of an altitude of approximately 1800 feet or above is coloured white and hence stands out distinctively against the green lower terrain as well as providing the obvious height cue of 'snow-capped peaks'.

2.4 Weather information

Through contacts at the Swedish Meteorological and Hydrological Institute (SMHI), the organization which is responsible for all weather forecast information across Sweden, data sets have been obtained which are similar to those provided to the Swedish Civil Aviation Authority for use by air traffic controllers. This information takes the form of geographically located weather forecast data giving the probability of certain weather conditions at each point in a grid across the region and with respect to altitude by flight levels. This information includes such features as predicted turbulence and icing dangers at the grid points and across a range of altitudes. We have incorporated this information within the 3D environment using the original grid cells and altitudes marking zones with significant risk (higher than a user-defined threshold) of these phenomena occurring. The danger zones are then represented in the display as translucent coloured blocks using textured surfaces to distinguish between the icing warnings (uniform blue) and turbulence warnings (patterned, white). These are clearly visible in the scene shown in Figure 3.

Other weather information has been incorporated during the most recent development phase with method for the representation of air pressure and wind speed being included through the use of pressure isobars (both as a single 2D layer and as a 3D isobar structure) and the inclusion of animated stream-particles (stream-line segments) which show wind speed and direction. Other, simpler representations of pressure and wind speed using cutting planes and vector plots have also been explored. Finally, to show air movement, we have introduced a method based on line integral convolution[8] (LIC) on a per-plane basis to show areas of strong air movement which the controller may wish to avoid. These weather representations are shown in Figure 3.

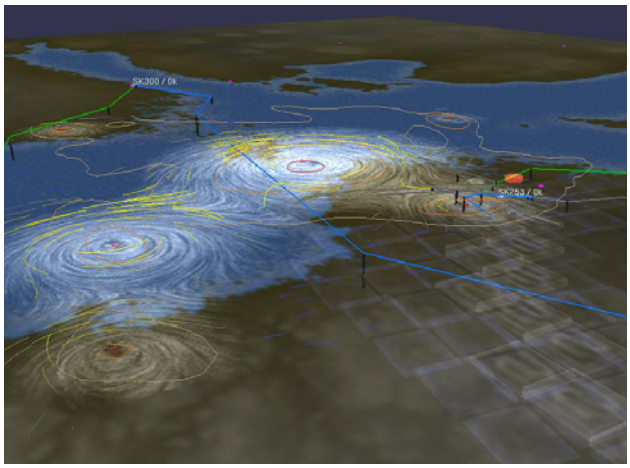


Figure 3: Weather representations in the current application showing pressure, stream particles and LIC representation of air movement throughout the supplied forecast data.

All weather information can be updated during execution of the application with updates in real time as new measured and forecasts data become available.

2.5 Interaction mechanisms

Within the workbench environment we make use of 3D interaction devices both to navigate in the scene and to select and manipulate flight information within the scene. The user's viewpoint is typically centred on a point of interest (initially located at the airport) but can be updated to centre around any position by simply pointing and clicking using a 3D wand pointer. The same wand device can then be used to control rotation of the camera around the view centre in two degrees of freedom and zooming of the view. The wand device can also be used to expand the height scale to emphasize the height of the 3D objects such as flight paths and weather information. In the latest version of the system this 3D interaction is supplemented by a powerful and flexible voice recognition system based on a discrete command set allowing the user to mix 3D interaction and voice control of the specific features of the system at will.

Once an appropriate viewpoint has been found, the user is able to select and manipulate the planned flight path of an individual aircraft. To overcome the frequent problems with overlapping flight paths, where different aircraft will share waypoints and paths between pairs of waypoints, the selection has been made on the basis of a specific flight. The controller selects the flight of interest using the wand pointer. This automatically highlights the flight path of that aircraft and renders the waypoints making up that path selectable by the user. The controller can then either move the selected waypoint or can insert and position a new waypoint between two existing waypoints.

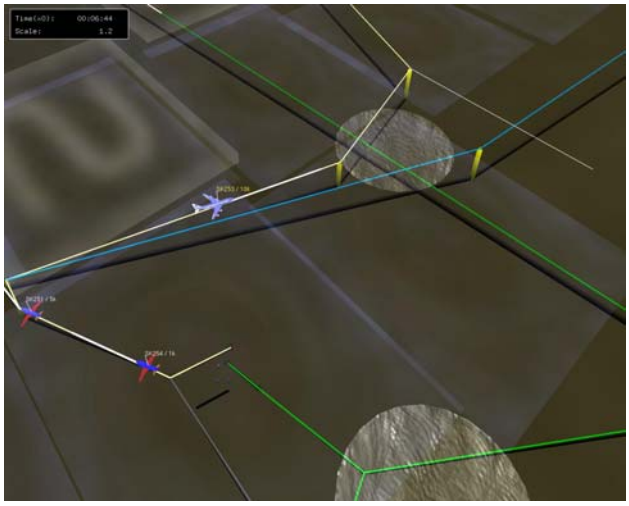


Figure 4: Having selected a flight, the 747 in the centre of the view, the user then selects the waypoint to reroute the aircraft.

The movement of a waypoint or the insertion of a new waypoint corresponds to the creation of a new 'virtual' waypoint, which is to say one which does not correspond to a real physical radio transmitter, and positioning it. This is a common practice in modern aviation. The positioning of the new waypoint, in all three dimensions, is carried out using the wand pointer before the waypoint is finally released. To aid the controller in selection the selectable objects: aircraft and the currently selectable waypoints, exhibit a strong localized 'gravity' which snaps the pointer to the object.

2.6 Implementation and performance

The program has been implemented using OpenGL for the rendering and CAVELib for the management of viewports and handling input from the keyboard and head and wand tracking devices. The flexibility of both these APIs makes the system extremely portable across platforms and the majority of the development has been carried out on Linux-based PC's while the system is targeted towards the semi-immersive workbench and VR theatre installations driven by an Onyx2 multi-pipe system.

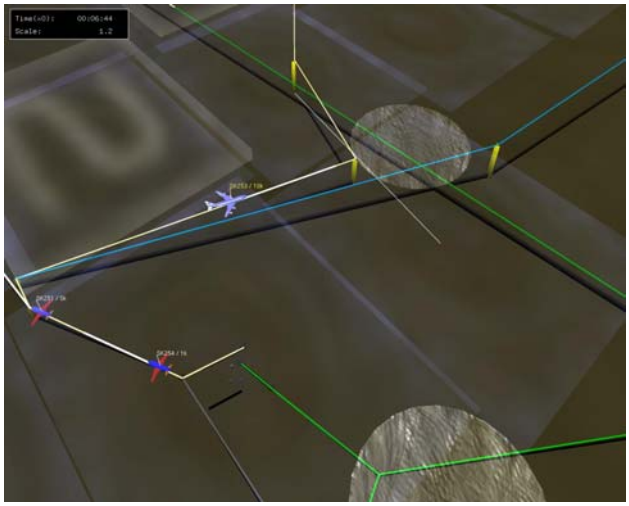


Figure 5: The selected waypoint is interactively rerouted such that the flight path bypasses the reported region of turbulence.

To make use of multiprocessing on the Onyx2, the system is implemented with each major functionality allocated to a separate process as shown in Figure 6. Keyboard input is handled through the application process and tracked objects (head and wand) are handled through the existing tracking management process. The 'Update' process controls the execution of the simulation, updating the positions of the objects within the scene. The 'Server' process is used to receive user input managing the representation of objects in the scene such as height colour-map information and other display settings. This originates from one or more text, graphical or voice clients, typically running on separate computers. The 'Sync' process ensures that the other processes act in a synchronized manner. The 'Draw' process(es) perform the rendering with one process required for each graphics pipe used in the current display.

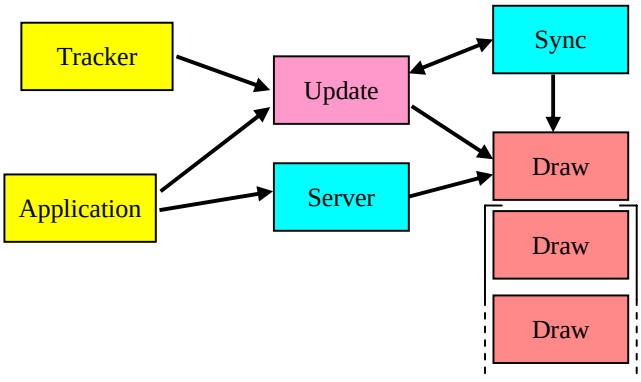


Figure 6: Process structure of the implemented system

Vertex arrays are used for most of the geometric objects in the scene. Many of the objects are also double buffered which allows coherent access by the various processes to the shared memory. 'Lightwave' object support is provided to permit the loading of the complex 3D objects such as the aircraft models. The Lightwave objects include the facility

to display additional information such as the current flight level by attaching colour information to certain parts of the aircraft such as the wings or tailplane. Rendering callback functions can also be defined for each individual model.

Each segment of a flight's planned route requires one quad for each 'curtain' segment and a GL line for the tube. The waypoints themselves are displayed using 3 quads per waypoint. Colour data to display height information on the curtains and waypoints is included through 1D textures. In the flight data set in the current work this produces a scene including approximately 1000 quads for the flight path information. The static terrain map used in the current scene comprises 4000 triangles.

On the Onyx2 system, driving either the Barco workbench through a single pipe or the three projector VR theatre through three pipes, the system typically gives a frame rate of approximately 20-25 frames per second. Recent experiments with a Linux based PC (Compaq 2.2GHz P4 with an NVidia Quadro4-900XGL graphics card) driving the Barco workbench has given rendering rates of approximately 40 frames per second.

2.7 VOICE RECOGNITION

The voice recognition system has been implemented using a small program developed using the Microsoft Speech Application Programming interface (SAPI) which provides facilities for the development of applications using discrete voice pattern recognition system. SAPI has been developed from the voice recognition research and development of Lernout and Hauspie which was acquired in a strategic partnership by Microsoft in 2000.

The voice recognition program we have implemented executes on a separate PC system and communicates with the main application using TCP sockets and sends simple text commands to control the features of the application. The voice command set available to the user, currently some 90 commands, has been carefully chosen to minimize the likelihood of misrecognition and, with careful training of the voice system and careful setup of the microphone headset and sound levels recognition rates of well over 95% have been routinely achieved making for a very relaxed working environment for the controller without recourse to a keyboard which has been essential until now.

The controller can use the voice system to switch on and off all of the visual display features of the system and to control some of the simpler navigation using command to rotate, elevate and zoom the camera as well as to focus on specific flights. The voice command "focus on flight S K 2 3 1", for example will cause the camera to smoothly move to the specific flight adopting a standard view on the flight selected. More complex navigational and interaction tasks such as selecting a flight and manipulating the waypoints defining its planned flight path still, however, rely on the use of a 3D wand. We are currently exploring methods and commands which might make even these functions available through the voice interaction system.

3. EVALUATION

Having developed and delivered the software system described here, it is currently undergoing evaluation with air traffic control staff at the Eurocontrol experimental research centre. These evaluations cover a range of properties of the system. Initially evaluations of the overall system are being performed which concentrate upon the reaction of the controllers to the 3D system and how clear and understandable are the visualization features which we have incorporated. This will provide us with guidelines for initial work to refine the system over the coming years.

In the longer term more detailed examinations of the ability of controllers to work with the system will be carried out. These studies will concentrate on the effectiveness of the 3D representation in exploiting the facility of human spatial memory rather than conceptual memory within this application area. Previous studies [2][3][4] have indicated that there is a definite difference in the way in which a user responds to three dimensional representations of data compared with two dimensional. When combined with visible geographical references[6][7], three dimensional data might be expected to be more readily recalled when cued with geographical information than conceptual information cued by data. If successful it is hoped that a transition to incorporate 3D displays in air traffic control may provide both an efficient working environment and may actually allow the controllers to manage a larger number of aircraft scattered over a wider area of terrain and altitude without any reduction in their ability to stay in control of the situation and, consequently, without any reduction in safety for flights and their passengers.

The evaluation of the effectiveness of the system in accessing and exploiting human spatial memory will be carried out in a further collaboration between researchers at Eurocontrol and at the University of Uppsala, Sweden.

4. FUTURE WORK

4.1 Planned new features

Initial evaluation work has provided feedback about the display and the representation which we will incorporate in future versions of the system. Primarily users have asked for additional information regarding orientation and locations cues. In addition to these changes we plan to introduce a number of new features into the project in future development phases as part of the ongoing exploration of new technologies within air traffic control.

4.1.1 Conflict detection and resolution

The principal concern in air traffic management is the early detection and resolution of potential conflicts. The displayed 'look-ahead' which we have included in these early versions of the system, showing where an aircraft will be some minutes into the future, provides some assistance here but is of very limited use as the problem is much more complex than we are able to include from our limited data.

Other researchers, working with Eurocontrol, are developing probabilistic approaches to conflict prediction and resolution using models of flight behaviour and we hope to include this work in future versions of the software. These detected potential conflicts will require the addition of warnings using visual, text and possibly audio cues to draw the user's attention to them. The use of spatially located audio cues may prove very useful since the user can potentially have relocated the camera such that the warning may occur outside of their immediate field of view.

4.1.2 Haptic interaction

The behaviour of aircraft and how they are affected by changes in their routing is extremely complex and typically modelled by a very complex set of equations. Selection of routes over long distances to minimize energy consumption and provide efficient use of the air space is an extremely complex problem being routinely solved in air traffic planning. When controllers wish to make navigational changes to avoid weather disturbances or reroute a flight to avoid a potential future conflict they are unable to take these very complex factors into consideration but, instead, must reply upon their experience and training to minimize the negative effect they may have in making a change. Using haptic interaction would provide a simple means by which the controller could be made more aware of the reduction in energy efficiency which their changes would cause. Ultimately the haptic device could provide a hard-limit to changes made, thereby preventing the controller from making changes which place the aircraft in danger of having to declare a fuel-related emergency.

4.1.3 Display and Interaction at airport

The system which has been developed so far is primarily targeted towards the management of aircraft in the relatively open air spaces between airports. Different problems appear when one considers the task of marshalling aircraft in the immediate environment around the airport where flights are much closer together and must be vectored into 'stacks' where they then hold awaiting final approach, landing and, finally, on-ground control as they approach the terminals.

When used in the management of flights in the large scale view, the application makes use of large aircraft models. These structures provide depth cueing information from their relative apparent size as well as providing a base onto which additional information can be placed. The real world scale of the objects displayed, however, can be measured in kilometres or even tens of kilometres. When close to the airport these very large objects obviously become a substantial problem and so we will make use of a rescaled environment with different representations of the aircraft so that they can be clearly differentiated. This revised application is planned for a future development phase of the project once we have received feedback on our current application from the evaluation processes.

5. CONCLUSIONS

The system developed provides a comprehensive interface for the air traffic controller to view, interpret and interact with complex flight data and supporting information affecting the controller's actions. The effectiveness of the display and its value in terms of presenting information to controllers is still being assessed by air traffic control staff and the design is being refined in accordance with feedback from that evaluation as it is received.

The project has enabled the consideration of a number of potentially useful technologies in this interesting and essential application area. It will also provide an excellent test environment for other new methods and technologies and it is hoped to continue this development and explore the application of more VR technologies in the future.

References

[1] Carla Kay Barlow, "Extra Sensory Perception". In *Air Traffic Technology International – 2000*.

[2] Andy Cockburn and Bruce McKenzie, "Evaluating the Effectiveness of Spatial Memory in 2D and 3D Physical and Virtual Environments", In Proceedings of *ACM C.H.I. 2002*, April 20-25, 2002, Minneapolis, Minnesota, USA.

[3] Czerwinski, M., van Dantzich, M, Robertson, G., and Hoffman, H. "The Contribution of Thumbnail Image, Mouse-over Test and Spatial Location Memory to Web Page Retrieval in 3D". In Proceedings of *INTERACT'1999*. pp163-170.

[4] Tavanti, M. and Lind, M, "2D vs 3D, Implications on Spatial Memory", In Proceedings of *IEEE Symposium on Information Visualization, 2001*

[5] Ronald Azuma, Mike Daily and Jimmy Krozel, "Advanced Human-Computer Interfaces for Air Traffic Management and Simulation", In Proceedings of *AIAA Flight Simulation Technologies Conference*, San Diego, CA, 29-31 July 1996, pp656-666.

[6] Desney S. Tan, Jeanine K. Stefanucci, Dennis R. Proffitt, Randy Pausch, "The Infocockpit: Providing Location and Place to Aid Human Memory". In Proceedings of *ACM PUI 2001*. Orlando, FL USA

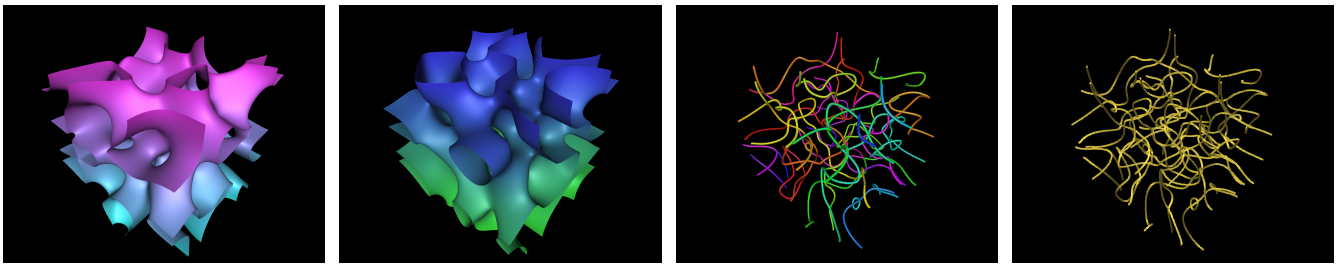
[7] Marlin, S.G., Tong, F., David, S. and Frost., B. "Testing Cognitive Maps of Immersive 3D Virtual Reality Environments". In Proceedings of the 4th Conference of the Australasian Cognitive Science Society, 1997.

[8] B. Cabral and L. Leedom. "Imaging vector fields using line integral convolution." In Proceedings of SIGGRAPH '93, pp. 263-272, Aug. 1993.

Extraction of Intersection curves from Iso-surfaces on co-located 3D grids

Patric Ljung <plg at itn.liu.se> Anders Ynnerman <andyn at itn.liu.se>

Scientific Visualization Group, Department of Science and Technology, Linköping University



(a) Surface of real field, 310627 triangles

(b) Surface of imaginary field, 308859 triangles

(c) Coloured intersection curves, 17104 line segments, rendered using 4-sided tubes

(d) Intersection points, 6615 points, rendered using tangent shaded points

Figure 1: The intersection of two iso-surfaces resulting in intersection curves and intersection points using the Marching Faces method. The curves represent nodal lines in quantum chaos. In general, bi-isolines are extracted from two scalar fields. The volume size is 128^3 for this chaos illustration, see section 8.

Abstract

This paper presents new methods for efficient extraction of intersection curves between iso-surfaces of any pair of co-located 3D scalar fields. The first method is based on the Marching Cubes algorithm which has been enhanced to produce an additional data structure that makes it possible to reduce the complexity of the general surface intersection extraction from $\mathcal{O}(N^2)$ to $\mathcal{O}(\sqrt{N})$, where N denotes the number of triangles in the arbitrary surfaces. The second method directly extracts the intersection lines based on finding intersection points on the faces of the voxels for two iso-surfaces extracted from a regular grid. A simple classification scheme is used for early termination of testing of voxels that are not intersected by both surfaces.

Also presented is an efficient method for fast curve generation through combination of line segments resulting from the explicit surface intersection method. An indexing structure is used to accelerate access and matching of intersection line segments to be combined into closed or open curves.

The presented methods have been used to identify and visualize nodal lines in 3D quantum and wave chaos data. These data are represented by a volume of complex values and a nodal line is a connected curve where the complex iso-value $z_{iso} = 0 + i0$. This type of chaos is believed to represent physical phenomena present in, for example, quantum mechanics, microwaves, fibre optics, and acoustics.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Boundary representations, Geometric algorithms; I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics data structures

Keywords: Intersection curves, Isosurfaces, Surface intersection, Feature Detection, Nodal lines visualization, Complex 3D fields

1 Introduction

Finding the intersection of surfaces is of importance in many application areas within mathematics and science. The meaning of the intersections depends on the application field. In most fields a topological sorting of the identified intersection segments is essential so that a set of distinguishable closed and open curves can be generated. The starting point for this investigation has been taken in the intersection of surfaces in chaotic quantum mechanical systems and results are presented in the context of the problems posed by this application. The work is focused on the intersection of iso-surfaces generated from co-located grids. Two alternative methods have been developed to identify the intersection of these surfaces. The first method uses explicitly created surfaces by means of iso-surface extraction algorithms, e.g. the Marching Cubes algorithm [Lorensen and Cline 1987; Montani et al. 1994]. This method is used when the surfaces as well as the intersection curves are of interest for visualization. With a minor enhancement the Marching Cubes algorithm enables efficient computation of the surface intersection with a complexity of $\mathcal{O}(\sqrt{N})$ where N is the average number of polygons in the surfaces. The general surface intersection problem is of complexity $\mathcal{O}(N^2)$.

As a second stage to the first method, line segments from the explicit surface intersection are combined into curves. For the type of data under consideration and for the purpose of this visualization, there is a need to identify the different curves. By colouring the distinct curves differently the interpretation of the generated images is improved.

The second method, named Marching Faces (MF), seeks to directly extract the intersections by implicitly considering the surfaces to intersect on the voxel faces in the sampled fields. The latter method avoids the memory consuming generation of polygons to represent the surfaces and only produces a significantly smaller point set for the intersection of the two surfaces and the voxel faces. It uses a simple voxel classification scheme to early detect voxels not intersected by both surfaces in order to avoid further testing. Both methods use efficient data structures for fast lookup of line segments, triangles, and intersection points.

The first method, using the Enhanced Marching Cubes (EMC)

algorithm, is preferred when the visualization of the surfaces is of interest as well as the intersection curves. The intersection algorithm then works with little additional cost. The MF algorithm is faster than EMC and thus preferred when surfaces need not be visualized. Both EMC and MF must visit all the voxels to ensure that all surface patches and intersections are detected, which constitute a fundamental problem since it grows with the volume size. Many methods have been proposed to speed-up iso-surface generation by using additional data structures, for example octrees [Wilhelms and Gelder 1992] or span-space [Livnat et al. 1996]. However, to the best of the authors' knowledge, in order to produce these structures, all voxels must be visited. For interactive pipelined processing of time-varying data sets, such pre-processing is not beneficial unless the accelerating structures can be used repeatedly. Or the following process otherwise would be of a complexity order larger than that of the volume traversal. Volume traversal is, in most cases, the most resource consuming operation but cannot be avoided in the proposed context of interactive pipelined processing of time-varying data sets, i.e. computational steering or visualization of raw, large scale, time-varying data sets.

2 Related work

Several methods for finding the intersection between surfaces have previously been proposed, in particular, the intersection of parametric surfaces have attracted extensive work [Patrikalakis 1993; Sabharwal 1994; Krishnan and Manocha 1997].

The general surface intersection problem is, in the naïve approach, an $\mathcal{O}(N_1 N_2)$ problem, or $\mathcal{O}(N^2)$ if $N \simeq N_1 \simeq N_2$, that is, all elements (polygons) must be tested against each other. By using subdivision methods or hierarchical methods it is possible to avoid intersection testing for patches of surfaces whose bounding volumes do not intersect. Extensive work has been dedicated to efficient extraction of iso-surfaces [Wilhelms and Gelder 1992; Livnat et al. 1996].

Previous work related to the Marching Faces algorithm is the 3D Marching Lines algorithm [Thirion and Gourdon 1996]. In this algorithm random seeds can be automatically placed in voxels of the volume. When a seed detects an intersection curve passing through a voxel, this curve is traced. To ensure detection of all curves, all voxels can also be searched. Even though the The Marching Lines algorithm is similar to the approach in the presented Marching Faces method it differs in some key aspects. The 3D Marching Lines method traces a curve segment when it is encountered. The Marching Lines algorithm determines the intersection points on the edges of the polygons from the first surface by interpolation whereas the Marching Faces method solves the intersection of the voxel-face/iso-surface intersection lines. The MF method also uses a classification scheme to entirely eliminate setting up any surface/voxel-face intersection lines unless both surfaces intersect a voxel cube.

3 Definitions

A scalar field ψ is defined as $\psi : \mathbb{R}^3 \rightarrow \mathbb{R}$. A subscript notation is used to refer to other data sets or variables that are unique to a specific scalar field ψ_i. For instance T_1 is the triangle set for a surface generated from ψ_1.

The angle notation $\langle\, x_i \mid i = 0, 1, \ldots, 7 \,\rangle$ is used to denote a vector. A simplified notation $\langle\, x_i \,\rangle_N$ having $i = 0, 1, \ldots, N-1$ is also used. The cardinality $|S|$ of a set S is the number of elements in the set.

Set	0	1	2	3	4	5	6	7
Real	88.13	3.12	6.09	2.17	0.49	0.00	-	-
Imag	88.07	3.06	6.24	2.15	0.48	0.00	-	-
Lines	99.20	0.15	0.24	0.26	0.12	0.03	0.00	0.00

Table 1: Histograms over percentage of voxels with a specific number of primitives, triangles for the surfaces and line segments for lines. Results are based on a 128^3 cube, e.g. 2048383 voxels.

4 Optimized Surface Intersection

This section describes the Enhanced Marching Cubes (EMC) method, the following intersection algorithm, and the method to combine intersection line segments into topologically sorted curves.

Under the condition that iso-surfaces are requested, the information obtained through the process of iso-surface extraction can be reused to significantly speed-up intersection testing. A few key properties of the triangles generated from the Marching Cubes algorithm is observed.

1. Any triangle can be uniquely located inside only one voxel.
2. The number of triangles in one voxel is a small number between 0 and 5. The vast majority of voxels have 1 or 2 triangles if and only if a surface intersects it.
3. Triangles can be enumerated in monotonically increasing numbers in the voxel traversal order.
4. Most voxels have no triangles at all. For the case shown in table 1, it is found that 88% of the voxels are empty.

These properties are exploited to optimize the surface intersection task. In essence, a subdivided space based on the grid on which the scalar field is sampled is obtained. For each voxel, a maximum of five against five triangles need to be tested for intersection. A voxel without triangles from both surfaces can be directly skipped. Table 1 shows some statistics for the distribution of elements-per-voxel.

4.1 The Enhanced Marching Cubes algorithm

The Marching Cubes algorithm is a well established method for extracting iso-surfaces from volumetric data. Triangles produced are uniquely defined within one voxel. Thus, for any co-located volume, testing of the triangles can simply be done on a per-voxel basis. This can be done immediately by traversing the two volumes simultaneously or by storing an indexing data structure to be used in a second stage. The advantages with the latter method are: The two surfaces can be computed in parallel if multiple CPUs are available. The Marching Cubes code requires only a very simple and restricted enhancement to support this case. Multiple sets of surfaces can be intersected without repeated traversal of the volume data.

The Marching Cubes algorithm is enhanced to produce an additional data structure, a triangle index structure I_T with the same dimensions as the processed volume ψ_i. When the MC algorithm traverses the volume it occasionally produces triangles and stores them in a triangle list $T = \langle\, t_j \,\rangle_N$. For each processed voxel k, any new triangles are appended to the triangle list T. The Enhanced Marching Cubes version then also stores the new size $|T|$ of the triangle list in a triangle index table I_T, $I_T(k) \leftarrow |T|$.

After traversing the full volume, the triangle index structure I_T can be used to query the presence of triangles in a specific voxel and to retrieve the index of the first triangle in that voxel. For two voxels in traversal order sequence, $k-1$ and k the number of triangles n in voxel k is extracted from I_T by

Algorithm: SURFACEISECT
Input: $I_{T_1}, I_{T_2}, T_1, T_2$
Output: L, I_L

```
1   L ← ∅
2   for each voxel k do
3     if I_T1(k−1) = I_T1(k) ∨ I_T2(k−1) = I_T2(k)
4       then continue with next voxel
5     for i = I_T1(k−1) to I_T1(k) − 1 do
6       for j = I_T2(k−1) to I_T2(k) − 1 do
7         APPEND(L, T1(i) ⋂ T2(j))
8       end for
9     end for
10    I_L(k) ← |L|
11  end for
12  return L, I_L
```

Table 2: Pseudo-code for algorithm SURFACEISECT that performs the surface intersection operation.

$$n = I_T(k) - I_T(k-1) \tag{1}$$

with the following boundary conditions for I_T

$$\begin{aligned} I_T(k) &= 0 \qquad k < 0 \\ I_T(k) &= |T| \quad k \geqslant N_{\text{voxels}} \end{aligned}$$

After application of the EMC algorithm on two selected scalar fields the output is passed to the surface intersection method.

4.2 Explicit surface intersection

The key element of an efficient surface intersection algorithm is to reduce the number of (triangle) intersection tests. By using the triangle index structure I_{T_i} the proposed method effectively reduces the number of tests. Given the inputs I_{T_1}, I_{T_2}, T_1, and T_2, the algorithm SURFACEISECT is outlined in table 2.

For each voxel k the presence of triangles in both surfaces is determined. If a voxel contains triangles from both surfaces, each pair of triangles are tested for intersection. For each intersection a line segment is generated and stored in the line segment list, L. When all triangle pairs in a voxel have been tested the current size of the line segment list L is assigned to the line segment index table, I_L, similar to the triangle index table. The further use of this structure is described in section 5.

For each voxel tested, a maximum of 25 triangle pairs need to be tested for intersection, typically 2–3 triangle pairs are tested according to table 1. For two arbitrary surfaces it is shown this intersection method works in $\mathcal{O}(\sqrt{N})$ where N is the number of triangles in a surface, e.g. $N = |T|$, see appendix A for details.

5 Combining line segments

The triangle intersection algorithm SURFACEISECT generates a list L of line segments and a line index data structure I_L. The line index data structure I_L provides fast look-up ($\mathcal{O}(1)$) of the presence of line segments within a local neighborhood.

The algorithm COMBINELINESEGS (table 3) combines line segments into closed or open curves by joining end-points of line-segments and creating lists of points. Each list defining a unique distinguishable curve. This enables an improved visual cue to perceive the different curves by using separate material properties, e.g. colour, over unrelated rendering of all segments with one and the same material property.

By iterating over all the line segments, nearby line segments are selected from the indexing set I_L. For each pair of these nearby line segments the algorithm matches end-points and selects the closest point to join. Since two end-points of two segments will match exactly, within rounding error, an alternative approach could be to stop searching when a match below a small threshold is reached. However, since line segments can be located arbitrary close to voxel vertices, within rounding error, the implemented method finds the point with the smallest error even for such degenerate cases. A threshold is also introduced to establish a maximum distance to allow connection of two end-points. Thus, this constitutes a greedy algorithm that always picks the best/closest end-point.

Algorithm: COMBINELINESEGS
Input: L, I_L
Output: CS

```
1   U(∗) ← ⟨ −1, ∗ ⟩,   CS ← ∅
2   for each line segment i ∉ U do
3     i_F ← i,   i_L ← i,   m ← 1,   m′ ← 1
4     do   [Backward trace]
5       ⟨ i′, m ⟩ ← FINDCLOSESTPOINT(i_F, 1 − m, I_L, U)
6       if i′ ≠ NIL then
          U(i′) ← ⟨ i_F, m ⟩
7         i_F ← i′
8       end if
9     while i′ ≠ NIL ∧ i_F ≠ i_L
10    if i_F ≠ i_L then
11      do   [Forward trace (open curves)]
12        ⟨ i′, m ⟩ ← FINDCLOSESTPOINT(i_F, m′, I_L, U)
13        if i′ ≠ NIL then
14          U(i_L) ← ⟨ i′, m′ ⟩
15          i_L ← i′,   m′ ← 1 − m
16        end if
17      while i′ ≠ NIL ∧ i_F ≠ i_L
18    end if
19    APPEND(CS, LINESEGSTRACE(L, i_F, U))
20  end for
21  return CS
```

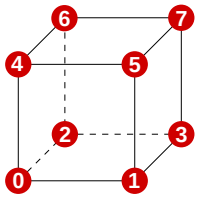
Table 3: Pseudo-code for algorithm COMBINELINESEGS.

The already used line segments are marked to prevent multiple connections and further matching. The used segments are marked in a set U. If segment a is linked to segment b the used set U is assigned $U(a) \leftarrow b$. After matching all line segments, U contains a linkage of points from the combined line segments.

The trace process is continued until the current curve reaches the boundary or connects back to itself, i.e. forms a closed curve. The algorithm first traces the first end-point backwards and identify the first segment index by i_F. If the curve is closed, this tracing ends when the initial segment is reached. Otherwise it continues to trace forward and updates i_L to indicate the last segment index. The traversal of L then continues to trace new curves among the line segments not already used. Note that the voxel is not marked as used, but rather the line segments are, since multiple curves can cross a voxel.

The algorithm finally outputs a list of curves, a curve list CS, each element of this list is a closed or open curve C. A curve consists of a list of curve points. A detailed view of the algorithm COMBINELINESEGS is presented in pseudo-code in table 3. The function FINDCLOSESTPOINT returns the closest matching point in the unused line segments within a local neighborhood. A line segment $L(i)$ has a start and end point, denoted $L(i)_0$ and $L(i)_1$, identified in the pseudo-code by variables m and m'.

Finally, the subroutine LINESEGSTRACE follows the linkage in the used segments set, U, beginning at the first segment index i_F and generates a list of curve points. The curve is defined as 'closed' if $i_F = i_L$, otherwise it is 'open'.



Face	No.
Left	0
Right	1
Front	2
Back	3
Bottom	4
Top	5

Figure 2: Voxel vertices and face numbering.

Algorithm: MARCHINGFACES
Input: V_1, V_2, iso_1, iso_2
Output: P, I_P

```
1   P ← ∅
2   for each voxel k do
3     b1 ←VOXELCONFIG(V1(k), iso1)
4     b2 ←VOXELCONFIG(V2(k), iso2)
5     f1 ← FaceTable[b1]
6     f2 ← FaceTable[b2]
7     f ← f1 & f2 & FACECHECK(k)
8     for each face j in f(j) ≠ 0 do
9       c1 ←FACECONFIG(V1(k), j, iso1)
10      c2 ←FACECONFIG(V2(k), j, iso2)
11      l1 ←SETUPLINES(V1(k), c1)
12      l2 ←SETUPLINES(V2(k), c2)
13      P' ←INTERSECTLINES(l1, l2)
14      APPEND(P, P')
15    end for
16    IP(i) ← |P|
17  end for
18  return P, IP
```

Table 4: Pseudo-code for algorithm MARCHINGFACES.

6 Implicit intersection of two iso-surfaces

The second method for finding the intersection of two iso-surfaces is based on directly extracting points of intersection from two co-located scalar fields. The creation of surface elements is avoided and computation is minimized by quickly discarding voxels and voxel faces where the two surfaces do not intersect. The points of intersection are defined by the intersection of two iso-surfaces and voxel faces. The authors have called this method Marching Faces due to its similarity to the Marching Cubes algorithm and its focus on intersection points on voxel faces.

The Marching Faces algorithm starts with two scalar fields ψ_1 and ψ_2 and traverses the voxels of these fields concurrently. The pseudo-code for MARCHINGFACES is found in table 4. Each voxel is classified by constructing a voxel configuration vector b (an unsigned byte). Equation 2 defines this vector, having the values d_i at the vertices and d_{iso} is the queried iso-value. See figure 2 for the numbering of vertices and faces.

$$b = \langle\, d_i < d_{\text{iso}} \,\rangle_8 \tag{2}$$

In the pseudo-code this operation is performed by the subroutine VOXELCONFIG.

To simplify the loop construction for handling of the boundary voxels where two opposing faces need checking, a function FACECHECK that returns a binary vector for ruling out testing of faces 1 (right), 3 (back), and 5 (top) for interior voxels.

Two voxel configuration vectors, b_1 and b_2, are evaluated from the two fields. Depending on the configuration, certain faces can potentially hold an intersection between the surfaces. A face with a mixed combination of vertices below and above the iso-value has the surface intersecting it. The possible voxel faces are encoded in a 256 byte array, **FaceTable**, for quick lookup of potential faces. By using the face-numbering in figure 2 it requires six bit codes. The bit-wise binary-and operator (&) on the face vectors from each voxel yields non-zero bits identifying faces that might hold a surface intersection point since the two surfaces intersect the same voxel face. For each of these faces the face configuration, c, is determined and encoded in four bits as illustrated by equation 3. This operation is carried out by the subroutine FACECONFIG.

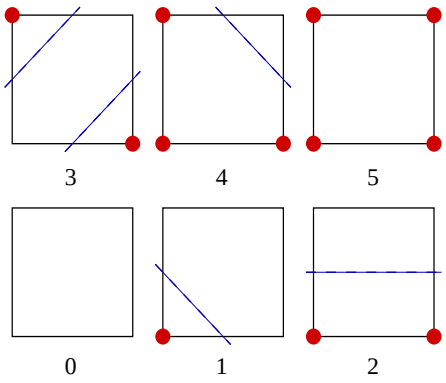


Figure 3: Face configurations. Red dots indicate a value below the iso-value. The blue dashed lines is the topological alignment of the iso-surface intersection.

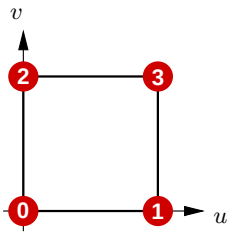


Figure 4: Numbering of the face's vertices and alignment on the parameter axis u, v.

$$c = \langle\, d_i < d_{\text{iso}} \,\rangle_4 \tag{3}$$

where the values d_i at the vertices i of the face are numbered as in figure 4. Figure 3 shows the unique six possible configurations of a face. By rotation all 16 combinations can be obtained. Each dashed line represents the topological location of the iso-surface. The points, p_j, of intersection on the edges with the corresponding iso-surface is determined by means of linear interpolation. Every pair of points, p_j and p_{j+1}, then identify an iso-line segment on the face. These lines are generated in the subroutine SETUPLINES. In face configuration 3, four intersection points are determined. This case represents an ambiguous configuration. The interpolated gradient orientation can be used to deterministically determine the best choice in this case.

By testing each pair of iso-line segments for the two fields on a face for intersection, one or two points on a face can be found to identify intersection points between iso-surfaces and the voxel face. Intersection curves thus pass through these points. The points are first defined in the local u, v coordinate system. The subroutine

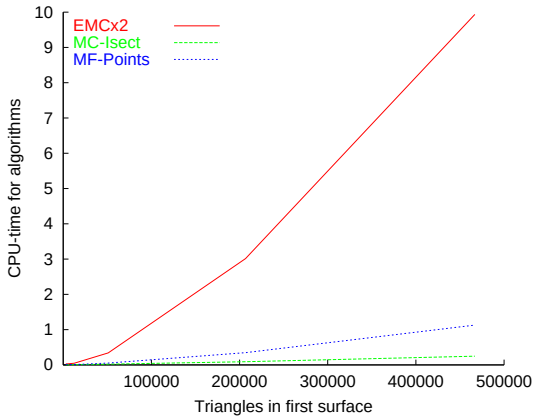


Figure 5: Execution times for the Enhanced Marching Cubes algorithm for two surfaces (EMCx2), Triangle intersection algorithm (MC-Isect), and the Marching Faces algorithm (MF-Points) as a function of the number of polygons in the first iso-surface. The variations of surface sizes have been generated by varying the sampling cube size. Tests were performed on a Laptop PC with Intel Pentium III CPU at 1 GHz and 512 MB of internal memory.

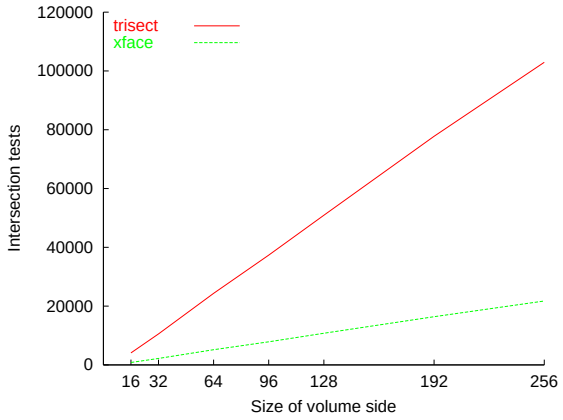


Figure 6: Performed intersection tests as a function of the side of the volume. The label 'trisect' refers to the triangle intersection algorithm using the triangles from two EMC passes, the graph specify number of triangle intersection tests. The label 'xface' refers to the Marching Faces algorithm, the graph specify the number of intersection tested voxel faces having a sign change at the vertices. Both methods are linear to the side of the volume, however, the MF method has a smaller constant in the ordo-notation making it faster.

INTERSECTLINES finds all line segments in l_1 intersecting with a line segment in l_2. The intersection points identified are returned in P', along with the computed tangential directions, as described below.

The gradient $\nabla\psi$ at u, v is estimated by bi-linear interpolation of approximated gradients at the vertices of the face. The gradients at the vertices are approximated by central difference, see equation 4, or a partially single sided difference at the volume boundary.

$$(\nabla\psi)_i(\mathbf{r}) = \frac{1}{2\delta}\Big(\psi(\mathbf{r}+\delta\hat{e}_i) - \psi(\mathbf{r}-\delta\hat{e}_i)\Big), \quad i = 1,2,3 \quad (4)$$

By taking the cross-product of the normalized gradients from the two iso-surfaces an estimate of the curves tangential direction is obtained at the point of intersection. For two almost co-planar surfaces the magnitude of the cross-product becomes very small, this measure can be used to place a weighting or confidence on the tangential direction. The estimated tangent can also be used as the direction of the curves' first order derivative for higher order curves. However, in the current implementation the tangent is only used for tangent shading [Zockler et al. 1996] for the rendered intersection points, see figure 1d.

7 Results

In figure 5 the timings for iso-surface and intersection lines extraction are shown for varying volume sizes with a fixed surface geometry. As can be seen, iso-surface extraction requires the most work. However, if iso-surface rendering is required, there is a small additional amount of work required to extract the intersection curves. However, the advantage of the Marching Faces algorithm is clear whenever surfaces are not required. Nonetheless, both the Enhanced Marching Cubes and the Marching Faces algorithms traverse all the voxels in the volumes and so these operations expand with the number of voxels. Profiling of the Marching Faces algorithm has shown that almost all execution time stems from traversing the volume and classifying each voxel. The computation has been reduced by using a simple caching scheme to save half of the bit-set of the voxel configuration when the algorithm advances to the following voxel. This scheme roughly halves the execution time of the Marching Faces algorithm.

In figure 6 the number of intersection tests versus the size of the cube side are shown. The graph clearly shows the linear relationship to the volumes' side for both intersection methods and that the more computational expensive part of the algorithm is bounded by linear complexity. Traversal of volumetric data is however a memory intensive task. By using blocking or bricking of the volume the data locality can be improved and thus yield higher cache hit-rates. See [Parker et al. 1998] for a discussion and implementation of an optimized bricking technique.

It can be concluded that both methods have an $\mathcal{O}(\sqrt{N})$ complexity for intersection testing ($N = |T|$). In terms of iso-surfaces, whose size depends on the side of the volume, the surface intersection methods are both linear with respect to the side of the volume. For the algorithm described in SURFACEISECT this reasoning holds if the iso-surface extraction is excluded. However, for the Marching Faces algorithm, the volume must be traversed. By expressing the algorithm complexity using s, the size of the side of the volume, we get $\mathcal{O}(s^3 + \sqrt{N}) = \mathcal{O}(s^3 + s) = \mathcal{O}(s^3)$.

8 Applications

Chaotic behavior is a phenomenon of importance in many fields of physics and has applications on multiple scales, including quantum mechanics, fibre optics, acoustics, and microwaves. The chosen application for this work is quantum chaos. Random superposition of monochromatic plane waves using the Berry wave function [Berry 1977] has been suggested as an approximation [Stöckmann 1999]. This wave function is then defined as a three-dimensional complex function $\psi : \mathbb{R}^3 \to \mathbb{C}$.

$$\psi(\mathbf{r}) = \sum_{j=1}^{N} a_j e^{i(\mathbf{k}_j \cdot \mathbf{r} + \phi_j)} \quad (5)$$

where a_j and ϕ_j are random numbers and $\mathbf{k}_j$ are random directions,

N defines the number of mixing states. In a bounded space like a cube, the function ψ must vanish on the boundary and equation 5 is rewritten to the form of equation 6

$$\psi(x,y,z) = \sum_j c_j \sin(k_{xj}x)\sin(k_{yj}y)\sin(k_{zj}z) \qquad (6)$$

where c_j are complex mixing coefficients, i.e. transformed versions of a_j and ϕ_j in equation 5. The sampling cube is slightly smaller than the domain of ψ in order to reveal the interior structure. See images in figure 1 for an example of a chaotic parameter region.

The properties of this function, e.g. where this function is zero $(0 + i0)$ defines the nodal lines of the wave-function. These nodal lines are of interest in applications such as electron transport in quantum mechanics. Nonetheless, the intersection of iso-surfaces has many other applications in the general fields of mathematics and physics.

The goal of this work is to provide fast extraction of these nodal lines so that interactive exploration of the properties of the nodal lines and corresponding electron transport can be conducted. For interactive exploration the authors take to mean that the entire process of computation of the complex field, extraction of nodal lines and visualization should be fast enough to provide acceptable update rates and user interaction.

9 Conclusions and future work

Two methods have been developed for efficient surface intersection of iso-surfaces, as embedded in volumetric data-fields. The algorithms have been tested on a complex dataset representing 3D chaos. By adding efficient data structures constructed on the fly, the performance has been improved for the intersection algorithm for triangular surfaces as generated by the Enhanced Marching Cubes algorithm. By using early classification of voxels in the Marching Faces algorithm the work is reduced for implicit intersection testing (roughly proportional to the side of the volume) to be almost negligible in comparison to volume traversal.

The indexing structures used for performance enhancement are currently implemented as volumes, which consumes internal memory. The authors consider these structures could be implemented primarily as two-dimensional structures, or by using hash-tables in order to reduce the memory requirement. It is also of interest to investigate the possibilities of caching obtained information for reuse in order to speed-up the intersection of consecutive data sets based on space-time coherence.

10 Acknowledgement

The authors wish to thank Karl-Fredrik Berggren at the Department of Physics and Measurement Technology, Biology and Chemistry at Linköping University for an interesting and encouraging problem to solve, which also happens to yield beautiful geometries. Financial support from the National Graduate School in Scientific Computing, the Swedish Research Council, and the Swedish Foundation for Strategic Research is acknowledged.

A Triangle Intersection Complexity

In this section a computational complexity of $\mathcal{O}(\sqrt{N})$ is deduced for two arbitrary surfaces. Assuming two arbitrary[1] surfaces with N_1 and N_2 triangles. The probability p_i for a voxel to contain any triangle in the triangle surface T_i is

$$p_i = \frac{N_i}{s^3} \qquad (7)$$

where s is the side of the volume. The probability p for two independent surfaces to intersect in a voxel is then $p = p_1 p_2$. For surfaces generated on a per voxel basis, the number of triangles in a surface is typically proportional to s^2, i.e. $N_i = c_i s^2$ for an unknown constant c_i.

$$p = p_1 p_2 = \frac{N_1 N_2}{s^6} = \frac{c_1 c_2 s^4}{s^6} = \frac{c_1 c_2}{s^2}$$

The number of triangle intersection tests t can then be expressed as

$$t = s^3 p = \frac{s^3 c_1 c_2}{s^2} = c_1 c_2 s$$

With $N_i = c_i s^2 \Rightarrow s = \sqrt{N_i/c_i}$ we rewrite t as

$$t = c_1 c_2 s = c_1 c_2 \sqrt{\frac{N_i}{c_i}} = c\sqrt{N_i}, \qquad c = c_1 c_2/\sqrt{c_i} \qquad (8)$$

The number of triangle intersection tests for two surfaces is thus proportional to $\sqrt{N}$, $(N \simeq N_1 \simeq N_2)$, given arbitrary surfaces. Two more or less identical surfaces leads to N intersection tests while two surfaces never intersecting yield 0.

[1] Arbitrary means a random distribution of the surface over the voxels in the volume and that the surfaces are not correlated.

References

BERRY, M. V. 1977. Regular and irregular semiclassical wave functions. *Journal of Physics A 10*, 2083–2091.

KRISHNAN, S., AND MANOCHA, D. 1997. An efficient surface intersection algorithm based on lower-dimensional formulation. *ACM Transactions on Graphics 16*, 1 (January), 74–106.

LIVNAT, Y., SHEN, H.-W., AND JOHNSON, C. R. 1996. A near optimal isosurface extraction algorithm using the span space. *IEEE Transactions on Visualization and Computer Graphics 2*, 73–84.

LORENSEN, W. E., AND CLINE, H. E. 1987. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of SIGGRAPH '87*, ACM Press, 163–169.

MONTANI, C., SCATENI, R., AND SCOPIGNO, R. 1994. A modified look-up table for implicit disambiguation of marching cubes. *Visual Computer 10*, 6, 353–355.

PARKER, S., SHIRLEY, P., LIVNAT, Y., HANSEN, C., AND SLOAN, P.-P. 1998. Interactive ray tracing for isosurface rendering. In *Proceedings of IEEE Visualization '98*.

PATRIKALAKIS, N. M. 1993. Surface-to-surface intersections. *IEEE Computer Graphics and Applications 13*, 1 (January), 89–95.

SABHARWAL, C. L. 1994. A fast implementation of surface/surface intersection algorithm. In *Proceedings of the 1994 ACM symposium on Applied computing*, ACM Press, 333–337.

STÖCKMANN, H.-J. 1999. *Quantum Chaos: An Introduction*. Cambridge University Press, Cambridge, UK.

THIRION, J.-P., AND GOURDON, A. 1996. The 3D Marching Lines algorithm. *Graphical Models and Image Processing 58*, 6 (November), 503–509.

WILHELMS, J., AND GELDER, A. V. 1992. Octrees for faster isosurface generation. *ACM Transactions on Graphics 11*, 201–227.

ZOCKLER, M., STALLING, D., AND HEGE, H.-C. 1996. Interactive visualization of 3d-vector fields using illuminated stream lines. In *Proceedings of IEEE Visualization '96*, IEEE Computer Society, 107–113,474.

Pressure Model of Soft Body Simulation

Maciej Matyka*
University of Wrocław, Poland

Mark Ollila †
Linköping University, Sweden

Abstract

Motivated by existing models used for soft body simulation which are rather complex to implement, we present a novel technique which is based on simple laws of physics and gives high quality results in real-time. We base the implementation on simple thermodynamics laws and use the Clausius-Clapeyron state equation for pressure calculation. In addition, this provides us with a pressure force that is accumulated into a force accumulator of a 3D mesh object by using an existing spring-mass engine. Finally after integration of Newtons second law we obtain the behavior of a soft body with fixed or non-fixed air pressure inside of it.

CR Categories: I.6.8 [Simulation and Modeling]: Types of Simulation—Animation; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: physically based modeling, animation

1 Introduction and Background

In this paper we present a model for three dimensional deformable objects simulation based on simple and fundamental physics principles.

We started our research in deformable object animation because of the complexity and cost of solutions with FEM, FEV (see [1; 15]) and LEM (see [9; 10; 11; 12]) methods. Some attempts with finite elements in real time simulations have been made (see [2]), however, the complexity of given algorithms is still high. Also similar problems appear with the Green function approach (see [6]) - complexity of implementation is high. Fast realtime pre-computed data-driven models of interactive physically based deformable scenes were proposed also in [14]. Methods using Navier-Stokes equations for Soft Bodies have been presented in [3]. The authors use Navier-Stokes equations for compressible fluid to compute properties of compressible fluid enclosed in a mesh. The model gives good soft behavior but solution speed is not efficient to achieve good results for real time animation.

In this paper, we introduce a novel idea of using ideal gas law in calculating pressure force. For pressure calculation we use a simple ideal gas state equation. Using the ideal gas approximation results in fast and physically correct animations.

*e-mail:maq@panoramix.ift.uni.wroc.pl
†e-mail:marol@itn.liu.se

The model which we introduce, is in a fact described by two equations and be very easy implemented into existing spring-mass systems by adding one additional force - the pressure force.

Figure 1: Real time animations created using presented model. Each ball object contains: 320 faces (162 vertices). Both examples run realtime on 800Mhz Duron processor machine (approximately 40-50fps, depending on collision detection configuration).

1.1 Particle System

Let us consider the governing elements of simple Spring-Mass (SM) model (see [4]. We will show how to expand it to simulate a 3D soft body with deformations. A simple SM engine contains a couple of obvious techniques, with the most important features to work with is a simple particle system which uses simple physics (Newton Laws), compute forces (Gravity, Hooke's linear spring force with damping) and makes numerical integration of one equation[1]:

$$\vec{F}_i = m_i \cdot \frac{\partial^2 \vec{r}_i}{\partial t^2} \tag{1}$$

2 Pressure Based Method for Soft Bodies

2.1 Method

The method presented in this paper is a soft body model based on classic cloth simulation (see [13]). Applying a wind force to cloth simulation results in very nice, good looking behavior.

[1]Where i indexes particles.

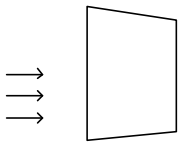


Figure 2: Cloth rectangle with fixed edges.

Lets say that we are presented with the situation depicted in figure (2). A cloth rectangle with fixed edges is placed, where wind, a force vector, is normal to initial surface of rectangle. What we obtain from the simulation like this is deformation of cloth under wind force.

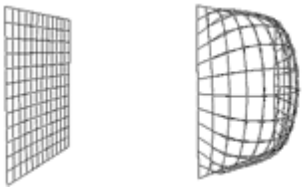


Figure 3: Cloth deformation under wind force.

Observing figure (3) it becomes apparent that it seems to be very similar to part of a deformed three dimensional object. Evolving the model further, we can "close the object" and put the "wind source" within it. Let us define model of shape as sketched in the figure (4).

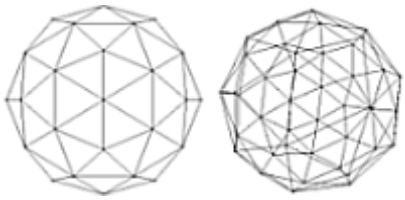


Figure 4: Triangular shape model with both solid and wireframe views presented.

Now let us imagine that we have some kind of "wind source" inside of that shape, which will introduce that nice deformation effect from figure (3) but in all directions. This is the basic idea of the implementation in this paper.

2.2 The Pressure Force

The simplest definition of pressure says that pressure is a force, acting on incremental surface elements and is parallel to normal vector of the surface. We would like to apply the force, which will act on our shape, and keep the shape geometry, but allow for possible deformations. Pressure, is a force that is always acting in a direction of normal vectors to the surface, so the shape will not deform significantly. So far we have only a definition of the pressure force vector, and we have to find some expressions for the force value. The expression for pressure in a specified point in the space acting on a surface is given by:

$$\vec{P} = P \cdot \hat{n} \left[\frac{N}{m^2} \right], \tag{2}$$

where P is a pressure value and $\hat{n}$ is normal vector to surface on which pressure force is acting. For calculating pressure force we have to multiply $\vec{P}$ by $A[m^2]$ - the area of the surface. That gives us pressure force expression [2]:

$$\vec{F_P} = \vec{P} \cdot A[N], \tag{3}$$

Now we will explain how to calculate P - the pressure force value.

2.3 Ideal Gas Approximation

In our model, we will use thermodynamic approximation known as "Ideal Gas Approximation" (see [5] for detailed description of that approximation). We can use this approximation because our interest is more in the macroscopic level effects of gas presence. At this level we can assume that in an object a gas of particles without interactions exist. We are only interested in the statistical properties (i.e. average momentum given from particles to the model surfaces in a specified incremental time).

The ideal gas approximation gives us simple relationship between pressure value, temperature of gas, and macroscopic volume of the body which can be expressed i.e. by the well known Clausius Clapeyron equation:

$$PV = nRT, \tag{4}$$

where P is pressure value, V is volume of the body, n is gas mol number, R is the ideal gas constant ($R = N_a k_b$, N - Avogardo number, k_b - Boltzmann constants), T is a temperature. From equation (4) we can easy get an expression for pressure if we know values of temperature and volume of the body:

$$P = V^{-1} nRT, \tag{5}$$

In the model presented in this paper we assume that $T = const$ and only the volume of a soft body changes. Specified assumptions will give us a very clear and easy to implement algorithm of pressure force calculation.

2.4 Algorithm

Before we dive into implementation details we will show the general algorithm of the presented solution. It will help clarify specific problems which appear during the implementation. The algorithm is based upon an existing particle spring-mass system with one modification - addition of a pressure force calculation. One computational step of the algorithm is as follows:

1 Loop over all particles:

  [1.1] Calculate and accumulate gravity and spring forces for all particles.

2 Calculate volume of the soft body

3 Loop over all faces:

  [3.1] Calculate pressure force acting on the face

  [3.1.1] Calculate field of the face

  [3.1.2] Calculate the pressure value

  [3.1.3] Loop over particles which define the face:

  [3.1.3.] Multiply result by field of the face and $\hat{n}$

  [3.1.3.2] Accumulate finally pressure force to the particle.

4 Integrate momentum equation

5 Resolve collision handling and response.

6 Move particles

[2] Where [N] means simply a force dimension.

3 Implementation

3.1 Calculate volume of the soft body

In Step 2 of the presented algorithm we have to calculate volume of the body. In the presented solution we used a variety of bounding objects to approximate that value. The implementation is well suited for non-complicated objects, and we will discuss later what kind of improvement can be done there. Bounding objects are a well known technique to speed up collision detection systems, and here we have used it to compute volumes. The type of bounding object depends strongly on geometry of simulated soft body (i.e. it is not very good to approximate ball by simple bounding box). In the model presented here we implemented three different bounding objects: bounding box, bounding ball, and bounding ellipsoid.

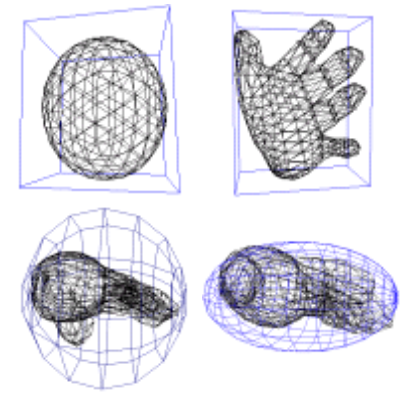


Figure 5: Three different types of bounding objects applied to the model.

We know calculating of volumes for bounding boxes, spheres and ellipsoids is fairly easy. For example, for the ellipsoid with r_x, r_y and r_z radiuses we have the expression for volume:

$$V_{el} = \left(\frac{4}{3}\right) \cdot \pi \cdot r_x r_y r_z \tag{6}$$

We use the term "bounding volume" for this technique of volume generation. Bounding volumes are not very accurate, but for a model, only the general change of body volume is needed. Of course for the hand model presented in the figure (5) a better approximation has been made with ellipsoid, and generally the ellipsoid has the best properties and is the most usable as a bounding volume.

3.2 Face field computation

Face field computation is quite simple, especially because of triangulated objects which we use as a models in the simulator. Simple algebraic vector multiplication of two edges is used here.

3.3 Numerical Methods

In the presented solution no special focus on numerical method has been done. We have used explicit Euler, Midpoint and RKIV integrators to integrate motion equations for every particle. It appears, that best choice is to use Mid Point algorithm, since we found that for some parameters configuration model is stable. All results presented have been calculated using 2^{nd} Mid Point scheme.

3.4 Summary of the implementation

Calculation of the body volume is one of the most important. After obtaining body volume, and the face field, we are able to calculate pressure force vector. Then basically accumulation of this force is performed.[3]

3.5 Collision Detection and Response

Existing techniques for collision detection and response (see [7; 8]) for deformable objects could be applied to described model of soft body simulation. For our purposes, for collision detection we use simple techniques of collisions with bounding objects. It is big simplification, but works very well for objects such as balls, cones, boxes, and other which are similar to the bounding objects. However, It is not very accurate, and future research needs to be performed here. Fortunately the results are rather good with the sample models used here.

The algorithm first detects which objects in the scene can collide. In the presented solution we used simple bounding boxes for first step of detection of possible contact.

Then a simple test of each point - bounding (ellipsoid/ball/box/other) object is applied. For each implicit formulation of bounding object, a simple value of the function is to be computed with the tested point. Tests have to be done for every point of one soft body with bounding objects of another soft body[4]. After that, if a point collides, we cut off $d\vec{r}$, the vector of movement (which obviously is to be given by the integration function to collision handling procedure). We have also implemented a simple iterative procedure which corrects the particle position if a particle intersects. That procedure checks if the point is inside of bounding object of the other soft body, if yes, then iteratively moves that point outside of the body as long as it will be outside of that bounding object.

When we recognize a collision, we implement a fairly simple collision response procedure. We divide the velocity vector of the particle into two velocities (parallel and tangent to collision surface) and we reflect the tangent part of it. Then both parts of velocity vector are multiplied by scalar coefficient to get effects of energy lost of particle during collision.

4 Results

In this section we present the results of the working application which was based on presented algorithm. In figures (1), (8) and (7) examples of working application are shown. All of these are taken directly from real time working animations. In the first two figures five bouncing balls with collisions, are placed in rotating box. The hanging ball with different pressures is shown in the next two figures. One ball object in the simulations contains 162 vertices and 320 faces. The third example in the figure (8) is a result of simulation with user interaction, where user can hit the simulated soft body (a hand in this case) with the rigid ball. The hand object contains ≈370 vertices with 736 triangles implemented into existing particle spring-mass engine. The simulations were computed in real time and run at (40-50 fps) on a Duron 800Mhz with RivaTNT2 graphics card.

In the figure (6) we present a plot of time needed for computation of pressure force. We prepared tests for an Athlon XP 1.8Ghz

[3] Please note that we have to apply force which is parallel to normal vector of the point (simple calculated by average of normal of its neighbor faces) and divide by integer value equal to number of faces, which contain that particle.

[4] Some tests can be done there, if checking first point-bounding box collision, then i.e. point-bounding ellipsoid is faster than checking only point-bounding ellipsoid intersection.

processor. The sketch shows times in *ms* for several different number of particles in the simulation space and average - trend line. We did not apply collision detection during these tests.

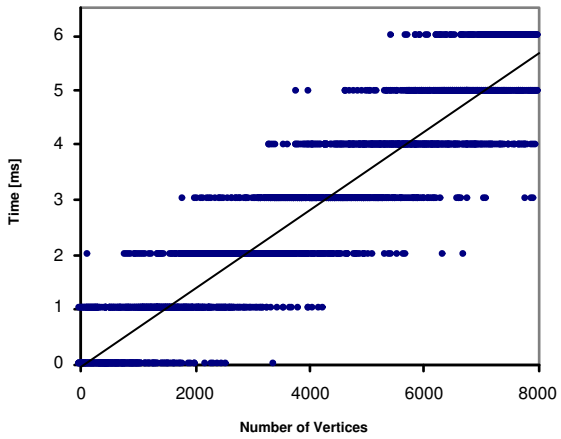


Figure 6: Time of pressure force calculation in *ms* in a function of number of vertices. Figure shows increasing of calculation cost, while number of vertices / triangles increases.

5 Conclusions and Future Work

Here we have presented a model on soft bodies using pressure. The most important advantages of the model is that it is fast (see figure 6), stable and can be easily implemented. However, there is significant future work to be performed. This includes work with more advanced volume calculation where we believe two methods are the most interesting to pursue. This includes development of some kind of bounding primitive subdivision algorithm to refine the volume representation. An alternative, is to compute volumes of the bodies with a Monte Carlo integration procedure. However, even though Monte Caro methods are accurate and converge to good results, they are rather slow and may not be suitable for real time applications. Future work will also include investigation into more complex objects and the suitability of this model with other forms of model representations such as Hypertextures.

6 Acknowledgements

The hand object presented in results section has been created by Mariusz Jarosz. Authors wish to also thank Jakub Kominiarczuk and Marcin Wojciechowski for support of the work. We are also grateful to an anonymous reviewer who motivated us to make the paper much better. The paper is the result of project work in the Modeling and Animation course at Linköping University.

References

[1] J. Teran, S. Blemker, V. Ng Thow Hing, R. Fedkiw, *'Cloth & deformable bodies: Finite volume methods for the simulation of skeletal muscle'*, Euro. Symp. on Comp. Anim. (SIGGRAPH Proc.), pages 68–74, 2003 G. Debunne, M.

[2] Desbrun, M. P. Cani, and A. Barr. *'Adaptive simulation of soft bodies in real-time'*, Comp. Anim., pages 133–144, May 2000

[3] Nixon, D. and Lobb, R., *'A fluid-based soft-object model'*,Comp. Graph. and App., IEEE , Vol. 22 Iss. 4, pages 68–75, July-Aug. 2002

[4] Witkin, A. and Baraff, D. *'An Introduction to Physically Based Modeling'*, SIGGRAPH Course Notes, 1993.

[5] Callen, H.B. *'Thermodynamics and an Introduction to Thermostatistics'*, 2nd edition, John Wiley & Sons, New York, 1985.

[6] Doug L. James, Dinesh K. Pai, *' Multiresolution green's function methods for interactive simulation of large-scale elastostatic objects'*, Jan. 2003, ACM Transactions on Graphics (TOG), Vol. 22 Iss. 1

[7] Steve Capell, Seth Green, Brian Curless, Tom Duchamp, Zoran Popović, *'Collisions and deformations: A multiresolution framework for dynamic deformations'*, Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, July 2002

[8] Matthias Müller, Julie Dorsey, Leonard McMillan, Robert Jagnow, Barbara Cutler, *'Collisions and deformations: Stable real-time deformations'*, Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, July 2002

[9] Cotin, S. Delingette, H. Ayache, N. , *'Real-time elastic deformations of soft tissues for surgery simulation '*, Visualization and Computer Graphics, IEEE Transactions on , Vol. 5 Iss. 1 , Jan. - March 1999, pages 62–73

[10] Balaniuk, R. Salisbury, K. , *'Dynamic simulation of deformable objects using the Long Elements Method'*, Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2002. HAPTICS 2002. Proceedings. 10th Symposium on, 24-25 March 2002, pages 58–65

[11] Costa, I.F. Balaniuk, R., *'LEM-an approach for real time physically based soft tissue simulation'*, Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on, Vol. 3, 2001, pages 2337–2343

[12] Sundaraj, K., Laugier, C., Costa, I.F., *'An approach to LEM modeling: construction, collision detection and dynamic simulation'*, ntelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on , Vol. 4, 29 Oct. - 3 Nov. 2001

[13] Xiaoming Wei, Ye Zhao, Zhe Fan, Wei Li, Suzanne Yoakum-Stover, Arie Kaufman, *'Natural phenomena: Blowing in the wind'*, Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, July 2003

[14] Doug L. James, Kayvon Fatahalian, *'Precomputing interactive dynamic deformable scenes'*, ACM Transactions on Graphics (TOG), Vol. 22 Iss. 3, July 2003

[15] Ralf Rabaetje, *'Real-time simulation of deformable objects for assembly simulations'*, Proceedings of the Fourth Australian user interface conference on User interfaces, Vol. 18, 2003

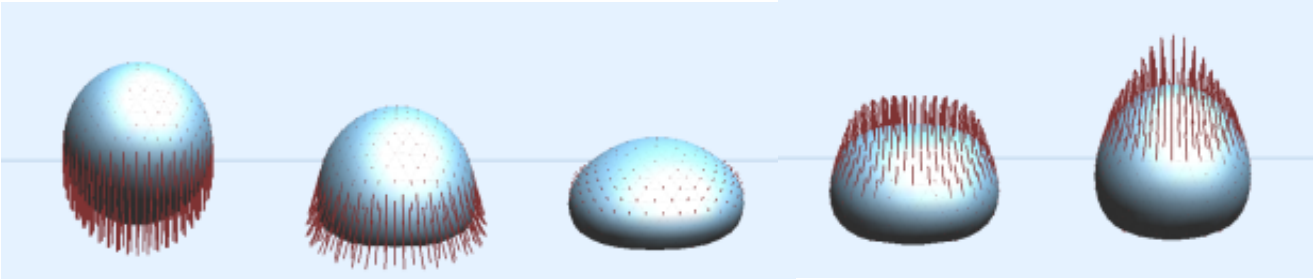


Figure 7: Bouncing ball with deformations. Visualization with velocity vectors for better information about physical property of moving object (i.e. momentum). Frame rate: 50fps. Details: 320 faces (162 vertices).

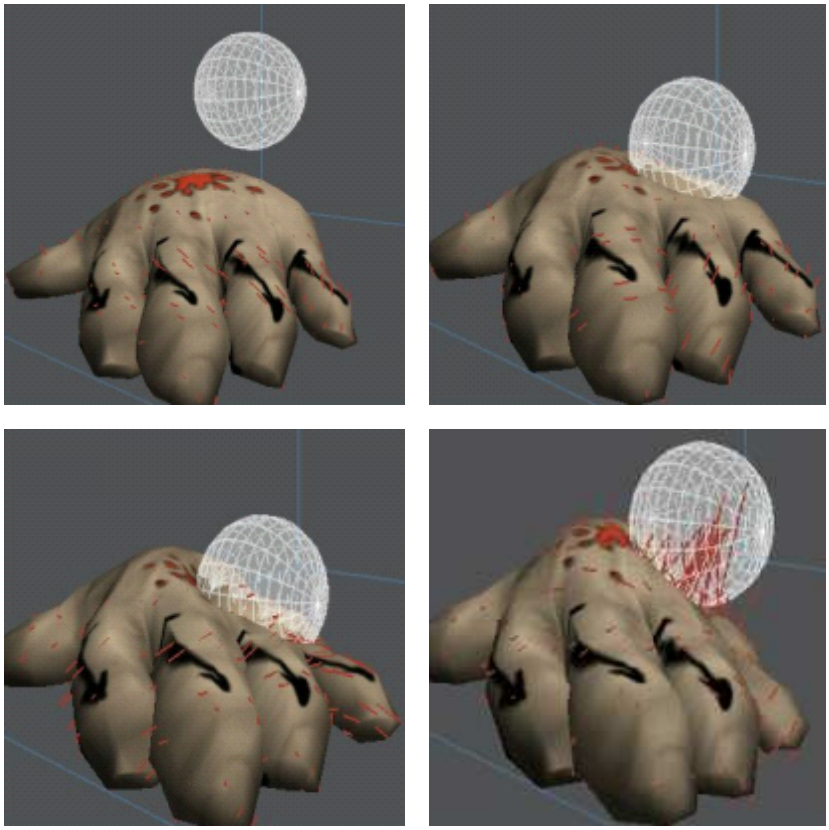


Figure 8: Hand object in gravity field with example of user interaction. Frame rate: 50fps. Details: 768 faces (386 vertices).

Synthetic Skies Using High Dynamic Range Images and Eigenskies

B. A. Olsson, A. Ynnerman and R. Lenz
Linköping University, Sweden

Abstract

This paper presents a method to render synthetic sky images corresponding to given weather conditions. The method combines artificial neural networks and principal component analysis to associate the appearance of the sky with the state of a weather parameter vector. This association is then used to generate artificial sky images corresponding to a given weather parameter vector. The proposed method has important applications for example in flight simulators and in the game industry.

The skies are represented by high-dynamic-range images which are able to store the dynamic properties of sky light. This representation can be used for global illumination in software packages such as Radiance to render scenes at arbitrary lighting conditions. The results show that, although the cloud details can not be represented by this method it is possible to distinguish between different weather states.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading and texture.

Keywords: Skylight, illumination and weather visualization.

1 Introduction

Lighting conditions and the appearance of the sky in outdoor scenes have a fundamental effect on the way human beings perceive an environment. Just by altering the light and appearance of the sky the same place can be perceived as frightening or comforting, cold or warm etc. In the fine arts and psychology literature the importance of the sky for human perception has long been recognized [DaVinci 1970]. This paper strives to encapsulate this importance in the context of computer graphics and presents a method to render synthetic sky images corresponding to given weather conditions. The method allows us to generate virtual worlds that exhibit a rich multitude of lighting conditions and sky backdrops. There are several possible applications of synthetic skies. One obvious application is flight simulators where it is important to generate realistic impressions of weather conditions. Another application is the entertainment industry where extreme weather conditions are frequently used in both computer games and in the film industry.

Our method of choice to represent sky light is based on high-dynamic-range images (HDR images) which can store the dynamic properties of the sky light [Debevec and Malik 1997]. The focus of the paper is to present a method to synthesize HDR images of the sky corresponding to a weather parameter vector. The method uses eigenskies, based on principal component analysis (PCA) [Joliffe 1980] of HDR photographs of a real sky, to compress the input data. Artificial neural networks (ANN) are then used to tie the values of a PCA coefficient vector to a vector of weather parameters measured at the same time as the HDR image was captured. The ANN can then, given any set of weather parameters as input, generate PCA components that are used to create a synthetic HDR image. If weather parameters from a forecast are used it is for instance possible to generate an HDR image corresponding to a future weather scenario.

The general field of weather visualization can be divided into two main categories. The first contains the more traditional approaches aimed at the creation of maps, in 2D and 3D, for weather forecasting. Trivis [Haase et al. 2000] which is used in the creation of maps for broadcast weather forecasts is one example of such a tool. To generate realistic images of an environment during given weather conditions is the task of the second approach. The analytic method of [Preetham et al. 1999] to render images of sky light is one example of this approach, which is able to render a general image of the sky using information about the viewer's location on earth, the position of the sun and irradiance values. The resulting image will be an average image for this time of the year, but no information about the daily weather is included and thus the dynamic change of the weather can not be seen in this image. Related methods using basis functions to fit simulation data have been developed by other authors. A series of Legendre basis functions was used by [Dobashi et al. 1995] and steerable basis functions were used by [Nimeroff and Rushmeier 1996] to fit sky luminance model data. Methods to synthesize clouds using real-time volume rendering taking subsurface scattering effects into account have been developed by [Riley et al. 2003]. The method described in this paper falls into this second category but will in contrast to other methods be able to render images which vary with the actual weather state.

To show the results of the method an application of the synthetic HDR images to define the lighting conditions in the rendering tool Radiance [Ward 1994] is described.

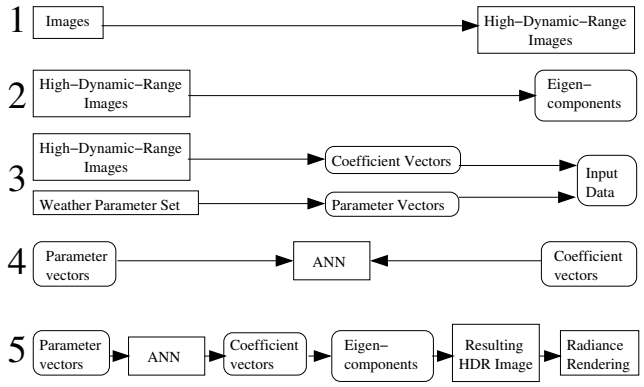


Figure 1: A diagrammatic representation of the main steps in the method. The method can be divided in five steps: Computation of HDR-images, computation of eigencomponents, data collection, training and synthesizing.

2 Method

Instead of explicitely defining the appearance of the sky in various weather conditions we collect images from different weather states. These images are then used to train the model. The method uses principal component analysis (PCA) to compress the data and learn the most important features of the images. The eigenface [Turk and Pentland 1991; McGuire and D'Eleuterio 2001] method inspired this work. The relation between the weather parameters and the image features is learned by ANN. We thus use ANN as a statistical method for function approximation. Our method is based on a combination of PCA and neural networks with supervised training.

In our approach we first capture a series of images with varying shutter speeds at each time interval (An image series can be seen in Figure 3). One series is used to construct one HDR image. The method used for computing HDR images is described in detail in [Debevec and Malik 1997]. The resulting fisheye images define the irradiance in every direction in Euclidean space.

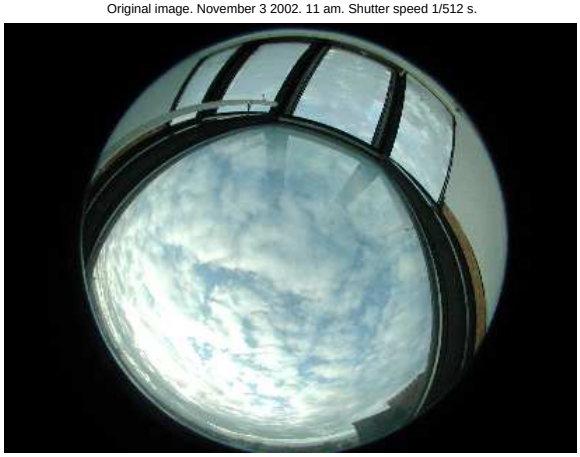


Figure 2: This is one example of the original fisheye images of size 1500x2000 pixels. The images used are captured of exactly the same scene by a camera located in a roof window. Some parts of the field of vision are obstructed. These parts are interpolated using radial symmetry.

2.1 Principal Component Analysis

Next we use PCA to decompose HDR images as follows: The pixels in an image are rearranged into a vector $\bar{x} = [x_1, x_2, \ldots, x_N]$ of length N. We assume that we have M such vectors. The transpose is denoted $\bar{x}^T$. The mean vector $\overline{m_x}$ is defined as $\overline{m_x} = E(\bar{x})$ and the covariance matrix is $\mathbf{C_{xx}} = E((\bar{x} - \overline{m_x}) * (\bar{x} - \overline{m_x})^T)$, where $E(.)$ means the expectation value of a stochastic variable. The eigenvalue problem is then defined $\mathbf{C_{xx}V} = \mathbf{VD}$, where $\mathbf{V}$ is a matrix with the eigenvectors as columns and $\mathbf{D}$ a diagonal matrix with the eigenvalues along the main diagonal.

For the resolution needed in this project the size of the covariance matrix becomes impractical. The size of the matrix can be decreased by using dimensionality reduction [Haykin 1999]. Let $\mathbf{Y}$ be the rectangular data matrix of size NxM. The matrix has the same height as the number of vectors (M). The eigenvalue problem, $\mathbf{C_{xx}V} = \mathbf{VD}$, can be written as

$$\mathbf{C_{xx}} = \frac{1}{N^2}\mathbf{Y^T Y} \Rightarrow \frac{1}{N^2}\mathbf{Y^T YV} = \mathbf{VD} \qquad (1)$$

The size of the covariance matrix is reduced from a size of NxN elements to a size of MxM elements by multiplying with $\mathbf{Y}$ from the left and changing variable $\mathbf{W} = \mathbf{YV}$.

$$\frac{1}{N^2}\mathbf{YY^T YV} = \mathbf{YVD} \Rightarrow \frac{1}{N^2}\mathbf{YY^T W} = \mathbf{WD}. \qquad (2)$$

The eigenvectors are the eigenskies. New sky images can be generated from them by linear combination. The weights are computed from weather parameters using ANN's.

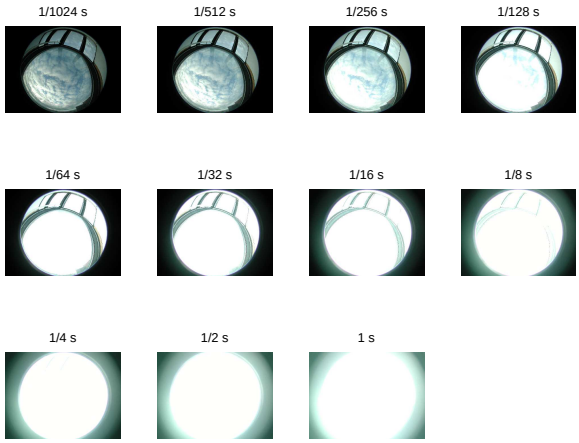


Figure 3: In order to compute a High-Dynamic-Range image a series of images is captured with varying shutter speeds. The first image is captured with a shutter speed of $1/1024$ s. For every image in the series the shutter speed is doubled until 1 s which is the last exposure.

2.2 Artificial Neural Networks

Artificial neural network is a collective term for a number of connected but in many cases very different methods. Neural networks originate from neuroscience as a historical method of simulating the brain neurons. This technology has found applications in diverse areas such as pattern recognition and signal processing. For a more detailed description of this subject see [Hagan et al. 1995] or [Haykin 1999].

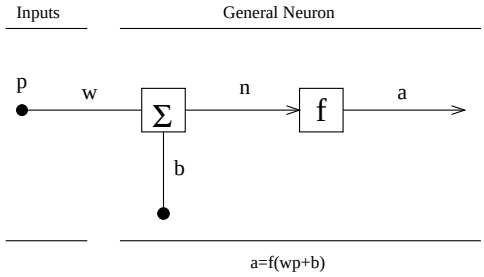


Figure 4: The single-input neuron. An activation function f is applied on the sum of the bias and the input p multiplies with a weight w.

ANN methods are used for function approximation in this paper. In the simplest form an ANN consists of one neuron (See Figure 4). The scalar input p is multiplied by the scalar weight w and then added to parameter b. The neuron output is calculated as $a = f(wp + b)$. Parameters w and b are adjusted by some learning rule. The activation function f may be linear or nonlinear. Examples of activation functions include the hard-limit activation function (also called the step function, $f(x) = \{0, x < 0; 1, x >= 0\}$),

the linear activation function and the log-sigmoid activation function ($f(x) = \frac{1}{1+e^{-x}}$). In a single layer network several neurons are all connected to the same input-signals. The mathematical formulation is identical to the single-neuron case, $\overline{a} = f(\mathbf{W}\overline{p} + \overline{b})$, but now $\mathbf{W}$ is a matrix and $\overline{a}$, $\overline{p}$ and $\overline{b}$ are vectors. A multiple layer network is built by connecting several single layer networks. In this paper we use two layers of the form

$$\overline{a} = f_2(\mathbf{W}_2 f_1(\mathbf{W}_1 \overline{p} + \overline{b_1}) + \overline{b_2}). \quad (3)$$

The first layer is called the hidden layer and the units of this layer are called the hidden units. The second layer is called the output layer. Matrices $\mathbf{W}_1$ and $\mathbf{W}_2$ and vectors $\overline{b_1}$ and $\overline{b_2}$ are the adjustable parameters in the two-layer case. Many different training methods exist but most are variations of the backpropagation algorithm [Rumelhart et al. 1986]. We used the Levenberg-Marquardt algorithm [Scales 1985], which is a variation of Newton's method.

The algorithm is provided with a set of examples of proper network behaviour:

$$(\overline{p_1}, \overline{t_1}), (\overline{p_2}, \overline{t_2}), \ldots, (\overline{p_x}, \overline{t_x}), \ldots, (\overline{p_n}, \overline{t_n}), \quad (4)$$

where $\overline{p_x}$ is an input to the network and $\overline{t_x}$ is the corresponding target output. The network output is compared to the target as each input is applied to the network. The network parameters are adjusted by the algorithm in order to minimize the mean squared error.

Data from two different types of data sets were used. *HDR fisheye images*, computed from eleven fisheye images with varying shutter speeds and *direct measurements of weather parameters*. Ten parameters were measured at the same time as an image was captured. The image database and the database of weather parameters were combined to train the model used to synthesize an accurate HDR fisheye image for a general set of weather parameters.

The method can be divided into five steps (See Figure 1). In the first step captured image series are used to compute high-dynamic-range representations of the sky light. The eigencomponents are computed from all the HDR-images in the second step. In order to compute well-defined eigenskies a large number of images at different weather conditions from a large time interval is needed. Every image is projected to the most important eigenskies and by this method a PCA-coefficient vector is computed. The vector represents a compressed version of the original image.

This vector is then combined with a weather parameter vector in the fourth step to form a data pair which is used to train the model. By using PCA the image data is compressed to a small number of coefficients. The first part of a data pair is an example vector of the input which in this case is a vector with the actual weather parameters at a certain time. The second part is the expected output at this input vector, a small number of PCA coefficients. We used twentyfour neural networks, one for each hour of the day, but the eigenskies were identical for all networks. In the fifth and final step, the synthesizing step, a weather parameter vector is sent to the neural network which generates a coefficient vector. This coefficient vector is used together with the eigenskies to synthesize a resulting image which is used in a Radiance rendering to define the sky light for this particular set of weather parameters.

3 Results

3.1 HDR images

An off-the-shelf digital camera (Nikon Powershot 990) has been mounted in a roof window. This camera captures an image series consisting of eleven images with varying shutter speeds (1/1024 s, 1/512 s, ..., 1/2 s, 1 s) every hour. An example image can be seen in Figure 2. Images from a time-period of seven months were used in this work. A C++ program to convert an image series to an HDR image has been developed. An image series of original HDR images is located in Figure 6. The original images were downscaled to a size of (150x200) pixels to lower the memory-demand. The ten most important eigenskies were used in the calculations.

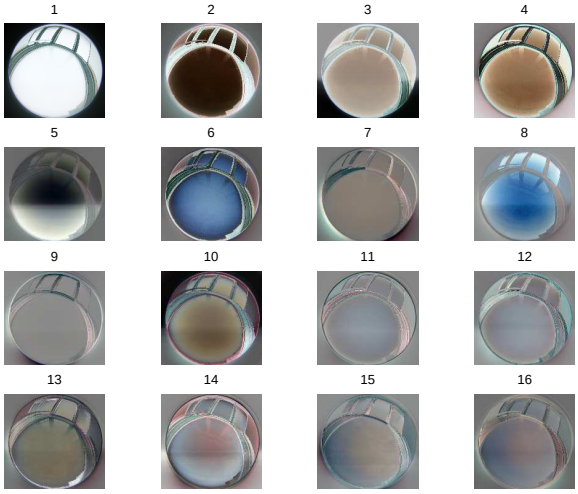


Figure 5: The sixteen most important eigencomponents - the eigenskies. Note the small artifacts in the middle of the images which are caused by reflections. The most important component, in the upper left corner, is a grey sky, which means that mostly the sky is grey.

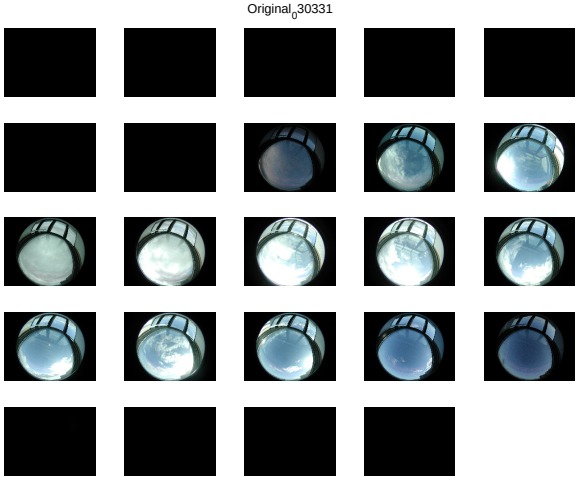


Figure 6: An image series of High-Dynamic-Range images. There is one image for each hour of the 24 hour day. This series was captured March 31, 2003.

3.2 Weather parameter vector

Direct measurements of the weather parameters were used for the training of the neural networks. These parameters were measured by an automatic weather station every hour at a position situated two kilometers from the camera. Ten parameters are measured of which temperature, pressure, date, time of the day, derivatives of pressure and temperature were used.

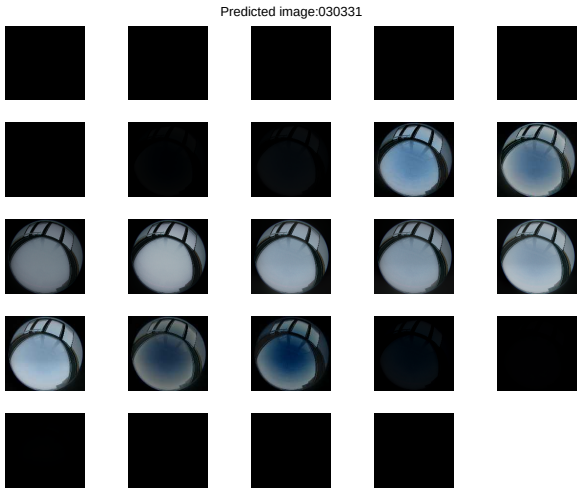


Figure 7: Images which have been synthesized using the method described in this paper. There is one image for each hour of the day. The neural networks were trained with seven months of image data and weather parameters. The weather parameters for March 31, 2003 were used to render these images. This day was not included in the training set.

3.3 The resulting images

All images were used to compute the eigencomponents (See Figure 5). The database of images was divided into twentyfour parts and each neural network was trained with data from one specified hour of the day. All networks were optimized using the available data. Most of the data was used, as a training set, to train the neural networks, but a number of data points were saved and used, as a test set, to verify the output result. The process to synthesize images was initiated by sending a weather parameter vector to the networks. A PCA-coefficient vector was predicted by one of the neural networks, depending on which time of the day the data point belonged to. The resulting image was computed by using this vector in combination with the eigenskies. This image represented a view of the weather at this specific data point. For every hour an ANN was trained to predict the eigensky coefficients from the temperature, pressure, time, time derivative of temperature, time derivative of pressure and date entries of the measurement vector. Twentyfour neural networks were used to simplify the training process of the individual networks. All images were transformed to coefficient vectors using the ten most important eigenskies. The two-level network used had a linear activation function on the hidden-layer and a log-sigmoid activation function on the second layer. All networks were trained until no further change could be seen. It was found that six hidden units gave a good compromise between flexibility and the risk of over-training. We have developed a software package in MATLAB [Olsson et al. 2003] that computes the eigenskies and trains an ANN to synthesize HDR-images. HDR images of the light field can then be synthesized from the artificial neural networks using weather data sets (See examples in Figure 7).

3.4 Results from Radiance

The synthesized HDR images can be used to define the light field in the *Radiance* [Ward 1994] system. HDR images of the sky light are synthesized by the model using weather parameter vectors as input. The resulting image is a map of the lighting conditions in the upper half sphere. Radiance is then used to render a general scene with the lighting conditions defined by the HDR-image. A resulting image series can be seen in Figure 8. The missing data was interpolated assuming radial symmetry.

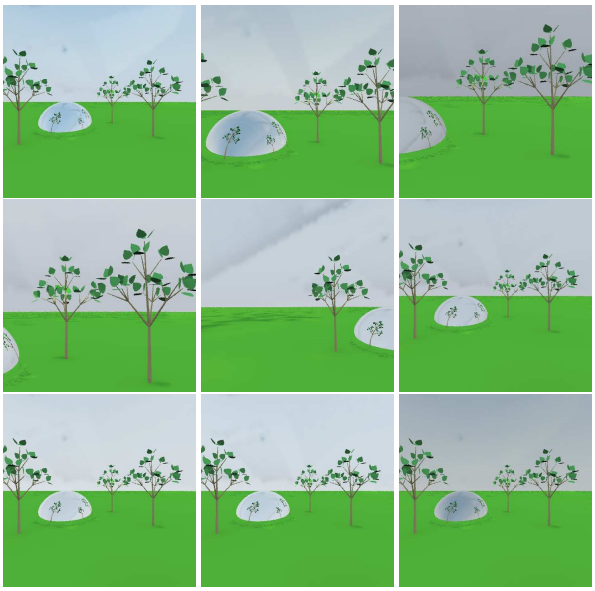


Figure 8: Examples of the result possible using the presented method in combination with Radiance. The High-Dynamic-Range fisheye images in Figure 7 were used to define the sky-light in a Radiance rendering. These images are the result from Radiance for 9 am until 5 pm. As can be seen the sky is varying during the day. Some small artifacts exist in the images due to internal reflections and interpolation errors.

4 Discussion

The synthesized HDR images have less detail than captured images due to the compression to a small number of coefficients. For example the cloud details are not represented correctly, but the main apperance of the sky is correct. The presented method should be seen as an alternative to the analytic method [Preetham et al. 1999] to synthesize sky light. While the analytic method of Preetham et al. is able to synthesize ab initio a realistic image based on average values of the irradiance depending on the position of the sun it is not able to produce an image varying with the weather conditions. The method presented in this paper is on the other hand able to synthesize an image, based on real photographs which varies depending on weather information for a specific instance in time. Although some differences in the intensity levels between the computed and the captured images exist, this method should prove to be a valuable aid in synthesizing sky images varying with the weather conditions. The obstructions in the images due to the camera location has been a major problem but this will be improved with an outdoor camera. A minor drawback is the loss of detail due to the time interval between the different exposures in the image series. No visible artifacts can be seen in the resulting images depending on this delay.

5 Future Work

Future work include increasing the number of parameters used in the neural network training to increase the precision of the prediction. Another future task is to generalize this method to synthesize sky images from parameters in weather forecast files and to combine this method with a method to compute realistic clouds from

parameters included in the forecast data sets.

6 Acknowledgements

The authors acknowledges the financial support of SMHI (Swedish Meteorological and Hydrological Institute) and NGSSC (National Graduate School in Scientific Computing). We thank SMHI for making the weather data available.

References

CATS, G., AND WOLTERS, L. 1997. The hirlam project. In *IEEE Computational Science and Engineering*, vol. 4, 22–31.

DAVINCI, L. 1970. *The Notebooks of Leonardo Da Vinci*. Dover.

DEBEVEC, P. E., AND MALIK, J. 1997. Recovering high dynamic range radiance maps from photographs. In *Proceedings of SIGGRAPH 97, Computer Graphics Proceedings*, 369–378.

DOBASHI, Y., NISHITA, T., KANEDA, T., AND YAMASHITA, H. 1995. Fast display method of sky color using basis functions. In *Pacific Graphics '95*.

HAASE, H., BOCK, M., HERGENROTHER, E., KNOPFLE, C., KOPPERT, H. J., SCHRODER, F., TREMBILSKI, A., AND WEIDENHAUSEN, J. 2000. Meteorology meets computer graphics - look at a wide range of weather visualizations for diverse audiances. *Computer & Graphics 24*, 391–397.

HAGAN, M. T., DEMUTH, H. B., AND BEALE, M. 1995. *Neural Network Design*. PWS Publishing Company.

HAYKIN, S. 1999. *Neural Networks*. Prentice Hall.

HIBBARD, A., AND SANTEK, D. 1990. The vis-5d system for easy interactive visualization. vol. 462, 28–35.

JOLIFFE, I. T. 1980. *Principal Component Analysis*. Springer.

MCGUIRE, P., AND D'ELEUTERIO, G. M. T. 2001. Eigenpaxels and a neural-network approach to image classification. *IEEE Transactions on neural networks 12*, 625–635.

NIMEROFF, J. DORSEY, J., AND RUSHMEIER, H. 1996. Implementation and analysis of an image-based global illumination framework for animated environments. *IEEE Transactions on Visualization and Computer Graphics 2*, 4.

OLSSON, B., YNNERMAN, A., AND LENZ, R. 2003. Skyvis, an application of matlab in meteorological visualization. In *Proceedings of Nordic MATLAB Conference 2003*, 295–300.

PREETHAM, A., SHIRLEY, P., AND SMITS, B. 1999. A practical analytic model for daylight. In *Proc. SIGGRAPH*.

RILEY, K., EBERT, D., HANSEN, C., AND LEVIT, J. 2003. Visually accurate multi-field weather visualization. In *Proceedings of IEEE Visualization 2003*.

RUMELHART, D. E., HINTON, G. E., AND WILLIAMS, R. J. 1986. Learning representations by back-propagating errors. *Nature 323*, 533–536.

SCALES, L. E. 1985. *Introduction to Non-Linear Optimization*. Springer-Verlag.

TURK, A., AND PENTLAND, A. P. 1991. Face recognition using eigenfaces. *J. Cognitive Neurosci. 3*, 71–86.

WARD, G. J. 1994. The radiance lighting simulation and rendering system. In *Proceedings of SIGGRAPH'94*, ACM Press / ACM SIGGRAPH, ACM, 459–472.

Examination of the possibility to use OpenSceneGraph for real-time graphics using Immersive Projection Technology

Linus Valtersson, M.Sc
CKK
Chalmers University of Technology
SE-412 96 Göteborg, Sweden
d98linva@dtek.chalmers.se

Abstract

This paper describes the examination of whether OpenSceneGraph (OSG) [1] can be used to render real-time graphics with Immersive Projection Technology (IPT). It starts by describing the systems used for graphics today and motivates why it is desirable to use OSG instead. After this, the examination itself is described. The result from this examination is that OSG can be used to render real-time graphics in a IPT-environment, but the task is not trivial. OSG alone is not enough to render graphics satisfactory, it requires an extra layer dealing with all VR-features included in the environment. To solve this, VR Juggler (VRJ) [2] was used on top of OSG. VRJ is not the only option to use together with OSG, but it was the one that proved to work most satisfactory. The combination of OSG and VRJ was used to implement an application displayed at the International Science Festival in Gothenburg 2003.

1 Introduction

In the field of Virtual Reality (VR) many new and interesting environments have been developed, this includes Head Mounted Displays (HMD:s) and different IPT-environments, which are the focus in this paper. An IPT-environment usually consists of four to six walls displaying stereoscopic graphics and some kind of tracking-device (FASTRAK, Flock-of-birds etc.) that keeps track of the user's and possible interaction devices' positions in space. An IPT-environment can be used for a number of different purposes, it can display for instance buildings, molecules and landscapes and it can also be used to create different kinds of learning aids or even games. Whatever it is used for, real-time graphics is essential.

Early applications mostly displayed a static scene like a building or a molecule. The user could possibly rotate and translate this scene but the scene itself stayed static. When the environment was developed further the ability to create dynamic scenes arose. For dynamic scenes some kind of programming is required to update the scene according to user-interaction.

The IPT-environment that will be the focus in this paper is located at Chalmers University of Technology in Sweden. At present there are two different systems that are used to create applications for this environment and these are DIVISION Mockup [3] and CAVELib [4].

DIVISION Mockup is a graphical user-interface used to create interactive scenes. The scenes are usually created by using some kind of modeling-tool, for instance Alias, and the scenes are later exported to DIVISION Mockup. In DIVISION Mockup the finished scene is composed and it is also possible to add some amount of user-interaction.

CAVELib is an OpenGL-based programming-interface. When using CAVELib the scene is created by using OpenGL-commands to draw the different parts of the scene. It is also possible to update the scene according to user-interaction for each frame.

These two systems differ in a number of ways. With DIVISION Mockup most of the work is modeling and than some programming might be required, whereas with CAVELib the entire work is programming. In DIVISION Mockup the ability to add dynamics is very limited, but in CAVELib on the other hand this is not a problem. The problem with CAVELib is that it uses OpenGL and standard C, which often requires an extreme amount of code (and time) in order to create a scene. As mentioned earlier, the ability to create dynamic scenes is very desirable but the only means to do so at present is to use CAVELib, which often takes to much time to use. To solve this problem a new system is needed. With the new system it should be possible to create interactive dynamic scenes without having to write an extreme amount of code.

OpenSceneGraph (OSG) is a newly developed graphics library that encapsulates OpenGL. As the name suggests it uses scene-graph-technique [5], which is much more smooth to use than pure OpenGL. OSG is an open-source project still under development but it already has a lot of users both in research and industry. Due to its popularity it seems like a good choice for application development, that is, *if* it can be used with IPT. The rest of this paper describes the examination of if OSG can be used in an IPT-environment.

2 Technical description of the environment

As mentioned in the introduction, the IPT-environment, which is the focus in this paper, is located at Chalmers University of Technology in Sweden. The IPT-environment at hand is a 3x3x3 meter TAN VR Cube with stereo projection on five sides (the four walls and the floor). The VR Cube is run by a Silicon Graphics Onyx2 Infinite Reality with 22 MIPS R10000 processors at 250 MHz, 2GB RAM and three graphics pipes. Users of the Cube wear Crystal Eyes shutter glasses, which are synchronized with the graphics. A Polhemus FASTRAK system tracks the position of the user's head and the position of the interaction device used. The interaction device depends on the system. With DIVISION Mockup a DIVISION 3D-mouse is used and with CAVELib a wand is used. The wand consists of three buttons and a joystick. It is also equipped with a sensor, in order for the tracking system to be able to locate it in space. There are also a Polhemus Stylus and 3BALL available.

The main environment for the examination was the IPT-environment described above. But for some early tests a ordinary desktop computer was used. This was a Silicon Graphics o2 with one MIPS R5000 processor at 200MHz and 256MB RAM.

The version of OSG used was OSG 0.93, which was the most recent release at hand. It should be noted that some facts about OSG mentioned in this paper are not true for newer releases.

3 The examination

OSG was first installed on a regular desktop computer. It was shortly discovered that to use OSG on a desktop computer was not very difficult. The distribution comes with a number of example programs, which are all well documented. To use OSG in the VR Cube on the other hand proved to be much more difficult. The OSG has classes that can be used to create a simple window to display a scene or the scene can be viewed in full screen. The stereo-graphics is not a problem since OSG has support for the stereo-technique used - quad-buffered stereo. The five projection walls on the other hand were a big problem. It was soon evident that OSG alone was not sufficient; some kind of extra layer was needed to handle the complex environment.

There exist a number of different libraries that can be used to deal with multi display systems and/or different interaction and tracking devices. Most of these have support for a number of different graphic libraries, such as OpenGL and OSG. Since OSG is very new there are not many libraries that support it, but two of them where found and these where OpenProducer [6] and VR Juggler (VRJ).

OpenProducer is also an open-source project and like OSG it is still under development. It was not very surprising that OpenProducer would have support for OSG, because it is developed by Don Burns, who has also developed OSG together with Robert Osfield. OpenProducer can handle multiple displays but one disadvantage is that it has no support for the tracking system used.

VR Juggler is also an open-source project, but unlike OSG and OpenProducer it has been around for a while and already exists in a 1.0 version and a 2.0 is on the way. VR Juggler is designed to be able to handle graphics and interaction in a number of different VR-environments. It can handle both the hardware and tracking system used.

The first combination to be tested was VRJ together with OSG because VRJ seemed a little more promising than OpenProducer. The idea behind VRJ is that it should be possible to run the same application in different environments without having to re-compile the application. This is possible due to the use of configuration-files. When an application is started the VRJ kernel gets the current configuration-file(s) and sets up VRJ accordingly. The VRJ distribution comes with a number of sample configuration-files. These include one file to run an application in "simulator mode" on a regular desktop computer and one for a four-sided Cube. The first goal was to create a simple program and get it running in simulator mode and then try to get it running in the Cube. To create a simple program was not very difficult; VRJ comes with a detailed step-by-step programmer's guide [7]. The first program just draw a cube in the VR Cube by using OSG and to run this in simulator mode worked just fine. To get the program up and running in the Cube, on the other hand, was not as easy. The sample configuration for a Cube that was supplied with the distribution was for a four-sided Cube using two graphics pipes and a Flock-of-birds tracking system. This differs quite dramatically from the system at Chalmers that has five walls, three graphics pipes and a Polhemus FASTRAK tracking system. A new configuration-file had to be created.

In order to be able to create a new configuration-file a deep investigation of the system was conducted. When the test-program was run it turned out that something was wrong in the configuration. The drawing did not work at all as expected and it did not follow the main shutter glasses, which is totally necessary in order to get a satisfying result.

Since the initial testing of VRJ was not exactly satisfactory, some time was spent testing OpenProducer. OpenProducer has been developed with a movie producer in mind. The idea is that a scene is created, by using for instance OSG, and then the number of cameras (viewports) desirable are set up (position, direction etc.). Whatever is caught on "film" is seen in the different viewports corresponding to the cameras. To test the combination of OpenProducer and OSG, the same approach as with VRJ was used. The first problem was known from the start, that OpenProducer had no support for input-devices or tracking-systems. Separate libraries would have to be created for this. Another problem was that there appeared to be a bug in OpenProducer regarding stereo-graphics, at least regarding quad-buffered stereo, which is used in the VR Cube. Due to all these drawbacks with OpenProducer, VRJ was given all the attention.

The initial testing of VRJ was not at all satisfactory. After some further tests it seemed like VRJ had problems with the tracking system used, Polhemus FASTRAK. To solve this, an alternative approach was used. When CAVELib is used, the values from the tracking system are read from a tracking-daemon called trackd. VRJ has support to use this as well and the configuration-file was changed accordingly. When this was tested the result was much better than the first time, but it was not satisfactory. This time, the drawing reacted when the user moved, but it was still not possible to see what was being drawn. The coordinate system appeared to the rotated and possibly translated.

After getting the drawing to react to user movements, it was time to make sure that the drawing got right. Something was obviously wrong with the coordinates, so the first step was to print out the coordinates for different positions in the Cube and compare these with the expected values. The expected value-ranges for the x-, y- and z-coordinates are –5…5, 0…10 and –5…5 respectively. Unfortunately, these were not the ranges that were read. The width of the ranges appeared to be correct, that is, all coordinates had ranges with a width of about 10, but the ranges themselves where not the expected. The x-coordinate seemed correct, but the y- and z-coordinates seemed to be shifted by about 7.8 and 1.6 respectively. After a deeper investigation of the configuration for the environment, these values turned out to be the transmitter offset, which was set to (-0.01,8.0,1.65).

The trackd daemon uses shared memory keys to store its data and these keys have to be known in order for VRJ to be able to find the correct data. These keys where easy to find in the old configurations and when they had been entered in the configuration for VRJ it was no problem to get the correct data.

When the data for the wand could be read correctly, the only problem that remained was the translated coordinate system. In the VRJ configuration files it is possible to add offset values for the different sensors, so it appeared as if the problem would be solved easily. However, the VRJ kernel did not interpret the values at all as expected and therefore the offsets were removed and the problem remained. The solution was to let the coordinates have the ranges they had and set the coordinates for the walls of the Cube accordingly. In other words, instead of adjusting the coordinate system after the Cube, the Cube was adjusted to fit the coordinate system. When the coordinates for the walls of the Cube had been set to fit the coordinate system the drawing appeared to work satisfactory.

After a lot more tests and a thorough investigation of the VRJ source-code the cause of most remaining problems where found. The cause was a bug in the VRJ 1.0 source-code that causes the matrix representing the sensor-data to be created in the wrong way. VRJ creates the matrix by applying the rotation angles in XYZ-order whereas it is supposed to be in ZXY-order to fit Polhemus FASTRAK. After correcting this, a re-compile of VRJ was attempted. But as it turned out, none of the machines available could understand the makefiles supplied with the VRJ distribution. Instead the solution became to create a new class that handles all contact with the trackd daemon and use this instead of the VRJ-classes. After this fix, everything appeared to work satisfactory.

4 Result

The goal was to examine if OSG could be used to render real-time graphics in an IPT-environment. As stated in the last part of the examination, a working configuration could be created in which OSG is used to render graphics. The task was not trivial however. Since OSG is “merely” a graphics-library and has no support for multiple displays, it alone can not be used with IPT. This is due to the fact that an IPT-environment uses multiple displays and furthermore the environment has a tracking system and several input-devises that OSG has no support for. The solution became to use an extra layer on top of OSG that handles the advanced features of the environment. This layer is at present VRJ, which can handle all advanced VR-features needed. Together they form a working platform, which has been used to create several sample applications and a larger application displayed at the International Science Festival in Gothenburg 2003 in collaboration with Swedish Television (SVT).

5 Future work

Even though a working platform has been configured, the examination is not complete. There are still questions that need to be answered. The current platform uses the wand for user-interaction. But as mentioned in the technical description, several other interaction devices exist. One question that needs to be answered is if it is possible to use any of these other devises.

When the basic examinations have been completed, it is time to look into how the platform can be improved to simplify application-development and fit the needs of the users of the system. Most users are not programmers but are more used to modeling. This implies that some kind of platform where the programming is minimized might be desirable. On the other hand, users who want to program their application from the bottom up should of course have the possibility to do so. With this in mind, a platform where the users can choose the level of abstraction depending on their needs and knowledge could be a good solution.

To develop applications using the current platform one needs to gain knowledge about both OSG and VRJ even though VRJ is mostly used just for getting tracking-information and to set the drawing-routine. Having to gain knowledge about both of these seems unnecessary even for the skilled programmer. To gain knowledge about OSG should be enough. This can be accomplished by developing a basic platform, which hides VRJ from the user. A skilled programmer can then use this basic platform to develop applications, while this platform can be developed further to get a higher level of abstraction for the not so advanced users. For the users who do not wish to program at all this can be further developed to create a graphical-user-interface similar to DIVISION Mockup.

Another approach is to perform further testing of the OSG+OpenProducer combination when OpenProducer becomes more stable. When the original testing was performed OpenProducer was in early alpha stage but now it has evolved to beta and the ambition is of course to be able to release a 1.0 version in a near future. It still has no support for tracking devices, but the data for these can be obtained by using the class that had to be written to deal with the bug in VRJ, and the displays can then be updated manually. This might improve performance because since OSG 0.94 OSG and OpenProducer are tightly integrated. The disadvantage is that the applications will be system-dependent since the code that handles the readout from trackd is system-dependent.

The same approach as for the OSG+VRJ combination could then be used. That is, to develop a platform with increasing steps of abstraction up to a GUI-interface.

So far the discussion has only been about graphics. But in DIVISION Mockup it is possible to use sound in the applications by using the Lake-system [8] available at Chalmers. With the Lake-system it is possible to get 3D-sound which can be "positioned" in space. To be able to use sound is a powerful tool, which makes applications more "alive" and therefore it is desirable to be able to use sound when using the new platform as well. No investigation at all has been made in this area so this is a big step to take.

Acknowledgements

This examination was carried out at CKK during the spring of 2003 and it would not have been possible for me to get this useful experience if Sven Andersson and Josef Wideström at CKK had not believed in me.

Finally I would like to thank Josef Wideström for his support during this examination and valuable help in writing this paper.

References

[1] OSFIELD, R. Introduction to the OpenSceneGraph; http://openscenegraph.sourceforge.net/introduction/index.html. 2003-09-14

[2] About the Juggler Suite of Tools. http://www.vrjuggler.org/about.php; 2003-09-14

[3] Division MOCKUP. http://www.division.com. 2003-09-14

[4] CAVELib overview; http://www.vrco.com/products/cavelib/cavelib.html. 2003-09-14

[5] BAR-ZEEV, A. Scenegraphs: Past, Present and Future. http://www.realityprime.com/scenegraph-php. 2003-09-14

[6] Introducing OpenProducer. http://www.andesengineering.com/Producer/index.html. 2003-09-14

[7] VR Juggler: The Programmer's Guide. http://www.vrjuggler.org/vrjuggler/docs.php. 2003-09-14

[8] Lake Technology Limited. http://www.lake.com.au. 2003-09-14

A Framework for Interactive Simulation

Dennis Gustafsson
Meqon Research AB
dennis@meqon.com

1 Introduction

Creating a large-scale simulator for interactive applications involves several different research areas, such as proximity detection, collision detection, dynamics solver, integration, etc. There are a large number of techniques available for each one of the specific areas, but very little research on how to combine algorithms into a general simulator, handling more than one specific task. Furthermore, there is also a need to combine several different simulation techniques in the same virtual environment.

2 Background

Meqon Research AB, founded 2002 from a diploma work at NVIS, Campus Norrköping, Linköpings Universitet, develop an SDK for interactive simulation targeted at both industrial simulation and interactive entertainment. The SDK started as a rigid body simulator but has now evolved into a general framework for any type of interactive simulation. The common bits and pieces of interactive simulation has been generalized and placed in a framework, so that new simulation techniques can be added very easily.

3 Framework

The framework relies on an interaction model, which is used to model interactions between various types of entities in a simulation. Interaction may be anything from collision detection and hinge joints to behavioral animation or interacting smoke. The interaction model is used to model pair-wise interaction between two types of entities. Interactions are implemented as classes in C++, making it easy to inherit existing interactions and create new ones.

The framework is used today primarily for rigid body simulation, but it has also been used for particle systems (Figure 1), cloth, hoses, vehicle dynamics and behavioral animation. Current projects in Meqon include simulation of deformable objects, water surfaces (Figure 2) and the human body.

During the presentation we will discuss the common parts of large-scale interactive simulations, the interaction model and how it can be used to combine several simulation techniques in the same virtual environment. Examples will be given on how the framework is used in the SDK. Brief knowledge on interactive virtual environments, computer graphics and different simulation techniques in the audience is expected.

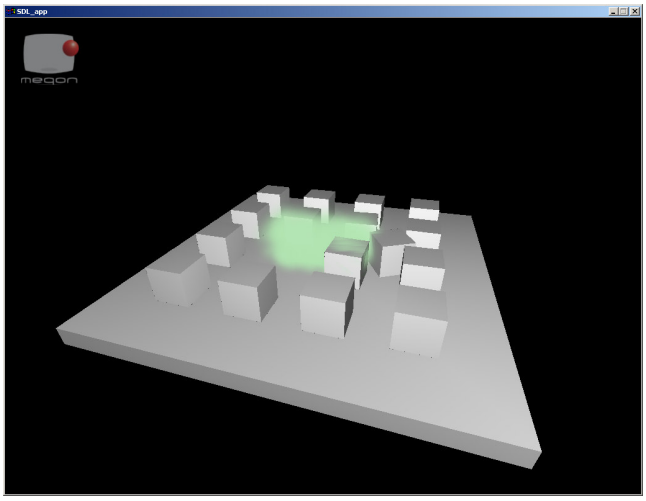


Figure 1. A particle system interacting with rigid bodies.

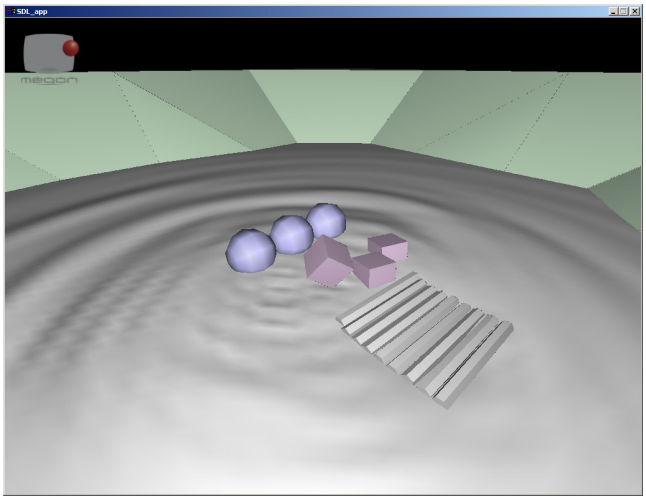


Figure 2. Rigid bodies interacting with a liquid surface.

Interactive Simulation of Granular Matter

Kenneth Holmlund[*]
VRlab/HPC2N, Umeå University

Andreas Lind[†]
Oryx Simulations AB

Rami Morrsy[‡]
Oryx Simulations AB

Abstract

We present work in progress on a method for interactive simulation of granular matter e.g. sand and gravel, and we show examples from a sample implementation in a vehicle simulator system (see figure below). Our model is based on a particle system with an attractive short-range pair potential, hard core repulsion, inelastic collisions and dissipation terms. Experiments have shown that granular matter in a gravitational field, or subject to external forces, is self stabilizing due to the build up of internal force columns. We use these and related results with heuristic reasoning to motivate how our model can produce plausible simulations for the dynamics of granular matter at an intrinsic intermediate length scale. This model system has also been integrated with a rigid-body simulation system.

Furthermore, we present an overview of the physics of granular material in general, and in particular an analysis of the necessary requirements on a model system for interactive and computer graphics intense applications.

We look at different norms for plausibility based on spatial and temporal considerations and also on the skill acquisition context of the vehicle driving simulator. Furthermore, we also analyze the real-time aspects and scalability issues of the problem.

Figure: Screenshots from the implementation of the presented simulation method in a vehicle simulator system illustrating interaction between the vehicle (rigid body system) controlled by the user, and the ground material (granular matter).

[*]e-mail: holmlund@hpc2n.umu.se, VRlab/HPC2N, Umeå University, SE-901 87 Umeå, Sweden
[†]e-mail: andreasl@oryx.se, Oryx Simulations AB, Uminova Science Park, SE-907 19 Umeå, Sweden
[‡]e-mail: rami@oryx.se, Oryx Simulations AB, Uminova Science Park, SE-907 19 Umeå, Sweden

A 6th order parallel CFD code based on an ENO-Padé scheme

Håkan Kihlström
hakki470@student.liu.se

Kristofer Lindberg
krili389@student.liu.se

Mark E. Dieckmann
mardi@itn.liu.se

Matt Cooper
matco@itn.liu.se

Norrköpings Visualiserings och Interaktionsstudio (NVIS), ITN, Linköpings Universitet

1 Introduction

Contemporary computer resources place the rendering of complex scenes for the professional film- computer games- and animation industry within range for physically accurate simulations. Models involving rigid body simulations, simulations of deformable objects, fluids and gases are increasingly being used to replace more traditional methods in animation and modeling. An important type of simulation is that for modeling fluids. Potential application areas are the simulation of water flow through the realistic modeling of flames to explosions. Simulations of liquid flow are an integral part of many recent movies and the real-time simulation of water is finding its way also into computer games like Half-life 2. However, most of these simulations solve the equations for incompressible fluids which excludes fluid compression and thus shock solutions and explosions. Without a physically accurate simulation we have to resort to procedural methods or geometric modeling to obtain animations of explosions. Explosions are however difficult to model since it is not generally possible to decouple the dynamical motion of the medium over multiple length- and time scales.

2 Exposition

Here we present our recent work on developing a 6^{th} order accurate simulation code that solves the Euler equations for compressible fluids which is discussed in [Wang and Huang 2002]. The Euler equations are a set of three coupled differential equations describing the transport of mass, momentum and energy in fluids. Specific properties of fluids are set by physical constants. The Euler equations are non-linear and the simultaneous occurence of processes on multiple time- and spatial scales complicates their analytic and their numerical solution. Typically, large spatio-temporal grids are required to resolve the large-scale structure of the fluid while ensuring that any waves with significant power that may be produced by physical processes have wave lengths larger than the simulation cell size. The required grid sizes pose challenges even to contemporary computer hardware. Our code is thus custom-written for parallel computers using the Message Passing Interface (MPI). The scheme also uses a fully implicit time discretization which will set more relaxed conditions on the time step chosen for the simulation. Further more we will discuss physical results and show simulation benchmarks obtained on Monolith, the massively parallel Linux cluster at Linköpings National Supercomputer Centre (NSC).

3 Results

To describe the performance of our numerical fluid solver we present a test case. The example shows a simulation that uses 560 cells and is evolved over 2000 time steps. To test the physical accuracy and the stability we have chosen initial conditions that lead to a shock-solution. The results are shown in figure 1. The figure show the fluid density in the system as a function of space and time. We have initialized the simulation with a rectangular density profile that moves at a high speed to the left relative to the surrounding

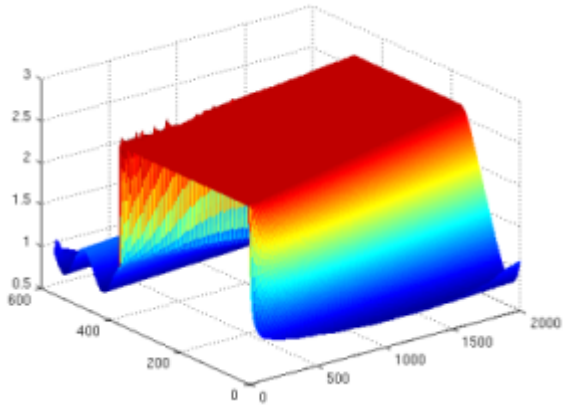


Figure 1: Plot of density over time.

medium. Ahead of the shock we have introduced some further sine-shaped oscillations. As you can see in the figure the backside of the density profile flattens. It reflects the suction force that the medium behind the density jump exerts on the fast-moving medium. The shock front shows a few wiggles but its slope constitutes a discontinuity without destabilizing. The ENO scheme has prevented the shock to smear out and the Pade scheme has provided excellent numerical accuracy across areas outside the shock discontinuity. This is in line with our expectations. We have also done efficiency tests on the code by comparing the simulations with identical number of grid cells and time steps for which we vary the number of processor nodes. If the code would scale perfectly we would expect a linear speed-up of the system with the number of processors. Our speed-up is not ideal. We obtain for example a speed-up of 2.25 for 8 processors instead of the ideal 8. These simulations do however only use a low number of cells per processor. This gives a problematic ratio of computations to communications. We can not yet extend the simulations to more realistic number of cells because we have not yet distributed the memory over individual processor nodes. We have also used an un-optimized matrix inversion scheme.

4 Future work

Currently our solver handles only one spatial dimension. As we commence our work of extending the solver to the fully three dimensional case we will also distribute the memory over the individual processor nodes and optimize the code's performance to allow for large simulation grids.

References

Wang, Z. Huang, G. P. 2002. An essentially nonoscillatory high-order Padé-type (ENO-Padé) scheme, J. Comput. Phys. 177, 37-58.

CGEMS - a refereed server to support the community of CG educators

Frederico C. Figueiredo
INESC/DEI/IST
Lisbon, Portugal
fepf@immi.inesc-id.pt

Dena E. Eber
Bowling Green State University
Bowling Green, OH
deber@bgnet.bgsu.edu

Joaquim A. Jorge
INESC/DEI/IST
Lisbon, Portugal
jorgej@acm.org

Lars Kjelldahl
Nada, KTH
Stockholm
lassekj@nada.kth.se

1 Introduction

CGEMS, the online Computer Graphics Educational Materials Source is a web-based groupware application that supports the submission, review, acquisition and archiving of curricular resources to be used by educators.

The rapid change of technology associated with computer graphics requires educators to learn new techniques and develop deeper insights on computer-generated images. As the core field becomes more mature, educators in all computer graphics disciplines have a greater need for high-quality curricular resources. By providing a repository for such materials, we can achieve a higher standard of teaching worldwide.

The purpose of CGEMS is to provide tools to support the community of Computer Graphics educators. CGEMS will allow their work to be appraised, assessed and made available to others through an online server for refereed educational content.

Although small systems and groups of people exist who are trying to address this issue, there is currently no centralized worldwide-refereed repository for computer graphics educational materials. Here we present a system that supports a way for educators to easily access quality course materials and for contributors to share and get recognition for their curricular innovations.

The Computer Graphics Educational Materials Source (CGEMS) is an online system available through the URL **`http://cgems.inesc.pt`**. The system includes a method for contributors to submit and editors to jury and control the quality of content to ensure sound and robust materials. The shape and components of CGEMS arose from fruitful discussions around, during, and after the Workshop on Computer Graphics Education [CGE02] held in Bristol, UK in July 2002. Figure 1 shows the CGEMS initial page.

Figure 1**. CGEMS Initial Page**

2 Editorial policy and submissions

Many discussions took place during and after CGE02 to shape the structure and policies of CGEMS. To serve the community of CG educators worldwide, we wanted to ensure (a) timely submission, (b) regular updates, (c) rigorous quality control, and (d) peer recognition. This led to establishing a journal-like system with several review cycles without a fixed deadline. This enables flexible review workflow and encourages timely updates of content. However, there will be regular calls for submissions possibly at the end of each academic semester in fall and spring. In this way, we hope to get notes, assignments, and examples from successful courses.

Ideally, we would like to have content organized in *course modules*, or a complete group of materials including notes, assignments, and examples that cover a specific subject. In other words, a module is a self-contained teaching unit including some or all of the above materials as part to an articulated whole. For example, a module could discuss shading networks for 3D modeling and the materials might include course notes, interactive demonstrations, assignments, and sample student work.

There are many quality-teaching materials that do not fall neatly into the module format, so the CGEMS server will also accept high-quality submissions that fall short of a full-fledged module in content, such as individual assignments or course notes. We are specifically looking for the following materials in order of preference:

(1) Complete Modules. (2) Annotated Course Syllabi
(3) Individual Lessons / Teaching Gems. (4) Annotated Problem Sets. (4) Lab Notes. (5) Annotated Student Work

We will accept the material in most common formats for which there are freely-available document readers.

While most if not all the materials currently assembled are written in English, we envisage and encourage both localizations and submissions in different languages, including Portuguese, German, French, Spanish, Swedish etc. While many educators in Sweden find it is still hard to cover all aspects of Computer Graphics in university courses, CGEMS offers a framework for using and providing material from and to other universities both within Sweden and worldwide, in an endeavor we hope will be useful to both educators as individuals and the community as a whole.

References

[Fred03a] CGEMS – Computer Graphics Educational Materials Server, Frederico Figueiredo, Dena Eber, J. Jorge, ACM/SIGGRAPH Educators Program, 27-29 Julho 2003, San Diego, EUA.

[Fred03b] A Refereed Server for Educational CG Content, Eurographics 2003 Education Program, Granada, September 2003.

[CGE02] Proceedings, Eurographics/SIGGRAPH Workshop on Computer Graphics Education, Bristol UK, July 2002. `http://virtual.inesc.pt/cge02`

Deformable objects in real-time with haptic feedback
Work in progress

Ola Nilsson* Mark E. Dieckmann†

NVIS, Department of Technology, Linköping University

1 Introduction

For realism in computer graphics today rigid body physics is no longer sufficient. Models that take into account soft objects and deformations give *better visual results*, and allows for *interaction* with *accurately modelled objects.*

Due to increased computer power, haptic feedback from soft bodies is emerging as an possible method of augmentation for information presentation.

We are currently developing a framework for simulating and visualizing deformable body physics using the finite element method (FEM), where the simulations include haptic feedback. This framework will be able to handle a large class of fixed topology objects described by elasticity theory.

2 Exposition

We have implemented a solver for simulating and interacting with deformable objects based upon FEM [Zienkiewicz and Taylor 2000]. The solver currently handles objects that are linear in material but nonlinear in geometry. The system exploits the sparsity of the FEM mesh and is coupled with highly optimized linear algebra subroutines for optimal performance. The solver uses the well-documented and widely used Newmark scheme. As of today we handle meshes of the order of thousands of degrees of freedom on a standard desktop computer and still achieve interactive refresh rates.

2.1 The numerical model

The system solves for accelerations of the displacements $\mathbf{u} = \{u,v,w\}^T$ in the linear matrix differential equation

$$\mathbf{M\ddot{u}} + \mathbf{C\dot{u}} + \mathbf{Ku} + \mathbf{f} = 0 \quad (1)$$

at each time step. We then track the surface nodal forces which are coupled back to the ReachIn desktop to give haptic feedback (see Figure 1).

The large uncompressed size of the matrices in equation 1 makes interactive solving impossible. However, with clever use of the sparsity structure real-time can be achieved. Especially when the solver is of the *direct* type, meaning that an inversion of one of the sparse matrices needs to be done. The sparsity structure of the inversion can be minimized, but this problem is NP-hard. We apply a Cuthill-McKee reordering scheme that approximates the optimal reordering. This transforms the size of the problem from $O(n^2)$ to $O(n^c)$ where c is close to 1.

We present the latest benchmarkings of our solver and discuss its components in terms of speed and accuracy, as well as physical realism. We will also present up-to-date results from the connection to the haptic system.

*e-mail: olani412@student.liu.se

†e-mail: mardi@itn.liu.se

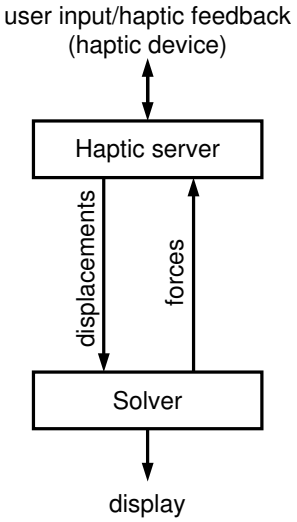


Figure 1: Connection of forces and displacements in the solver framework.

3 Results

We are currently connecting this solver to our haptics stations, ReachIn desktops with PHANToM haptics device. The haptic feedback is of today not fully implemented and under evaluation.

On an Athlon, 1.3 GHz, we solve a system with 1000 degrees of freedom at a refresh rate well above 100 Hz, and for 5000 degrees of freedom we get 20 Hz.

References

Zienkiewicz, O.C., Taylor, R.L. 2000. The Finite Element Method. Volume 1, Fifth Edition, Butterworth-Heinemann.

Wu, X., Downes, M.s., Goktekin, T., Tendick, F. 2001. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes, Eurographics Volume 20, Number 3.

Zhuang, Y. and Canny, J. 1999. Real-time simulation of physically realistic global deformation. SIGGRAPH99 Sketches and Applications.