2006

Interactive design methods in virtual reality with haptics

Andrew George Fischer
Iowa State University

Interactive design methods in virtual reality with haptics

by

Andrew George Fischer

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Mechanical Engineering

Program of Study Committee:
Judy M. Vance (Major Professor)
Ashraf F. Bastawros
James E. Bernard
Glen R. Luecke
Thomas J. Rudolphi

Iowa State University

Ames, Iowa

2006

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. INTRODUCTION

Engineering design is an evolutionary process involving the creation of new products and the refinement of existing designs to obtain an optimal solution. A large body of research exists on the optimization of existing designs with a number of mathematical and computational techniques. However, the earlier stages of design, from conception to first functional solution, has received much less formal computational aid. This research seeks to develop a design environment and methodology capable of aiding engineers in the earlier stages of design, when large changes are the norm and a number of different options are being considered.

Virtual Reality (VR) serves as the basis for this methodology. Working in a virtual environment fosters a natural interaction with digital products, encouraging rapid, large design changes and easy investigation of results. Interference with existing geometry is often more quickly remedied, and alternative solutions are easy to explore. Collaborative design is another great benefit. It's easy to have a number of designers and engineers working together within a large virtual environment.

This research ties the engineering analysis results to the shape design of products with real-time (roughly less than $1/15^{th}$ of a second) results in VR. Fast analysis approximations combined with stress interpolations let designers immediately see the effects of different product shapes. If analysis results are quickly approximated, more design options may be explored in a shorter time, making interactive design an important goal.

Goals

The work presented here seeks to improve on the interactive design methodologies

already in place. A virtual environment is generated using the VR Juggler software created at Iowa State University's Virtual Reality Applications Center. Within that environment users may load a model, interact with it, and deform selected portions. Making this interaction with geometry in the virtual environment as effective and intuitive as possible is very important. Virtual reality is only a real benefit if it gives the designer greater freedom than the traditional desktop computer interface.

Mesh-free analysis methods are used in place more traditional finite elements to avoid mesh distortion issues with deforming geometry. A custom analysis and solver was created to test the mesh-free methods with interactive design. However, the implementation is limited to a small class of problems and relatively untested. Implementing a more robust analysis with options for other material models and solution methods would be a great benefit to the existing methods.

Seeing analysis results, such as stress patterns, update in real time is one of the most important characteristics of interactive design. It provides immediate feedback to a designer about the effects of product changes. Linear Taylor Series approximations are used here due to their speed, at the expense of accuracy for large design changes. Using these approximations requires computing the stress sensitivities for shape design sensitivity analysis. This process can be computationally expensive, which takes up time and decreases the interactivity of the process. Improving the speed and accuracy of the existing stress sensitivity calculations is important to improve the interactive design experience.

Working with complicated and/or multiple objects in the virtual environment means a lot of information for a designer to process. The use of haptic or force feedback would provide an additional channel of information for the user about the design state. It also provides a convenient and more intuitive way to manipulate geometry. Implementing

a haptic device within the virtual environment could be a valuable tool for interactive design.

Organization of Dissertation

The above potential improvements were analyzed and implemented in a series of four papers that form the bulk of this dissertation. Four different areas were explored and actual improvements researched and implemented. Each major area forms the basis of one paper.

In Chapter 2 a brief history of the interactive virtual design methodologies is covered along with the state of the application. The most limiting factors for interactive design in the virtual environment are enumerated, specifically those related to working with the mesh-free analysis models. Solutions for these areas are implemented and the results and benefits are presented.

Chapter 3 presents the limitations of the custom mesh-free analysis tools created for immersive design. Alternatives are researched and compared with an emphasis on Open Source software solutions. The Tahoe analysis software, developed by Sandia National Labs, is selected as a replacement. Tahoe is integrated within the immersive design framework and compared with the custom analysis.

In Chapter 4 one of the most significant limitations to interactive virtual design is presented: computing the stress shape design sensitivities for use in the Taylor Series approximations. Computing these sensitivities takes significant time, and must be done often as the user alters different geometry in the environment. In addition the approximations are only as good as the sensitivities calculated. To remedy this review is made of the existing options for design sensitivity analysis. A discrete derivatives technique with exact numeric differentiation is selected, implemented, and tested against the existing finite differences technique.

Chapter 5 presents the addition of haptic or force feedback to the virtual design environment. Different haptic devices are compared and a large PHANTOM arm is selected. The integration of networked haptics with the existing application is then explained. Presenting haptic feedback to a designer based on stress contours is not a straightforward task, so several different techniques are compared. A small pilot study is designed and run to help gain an understanding of the benefit haptics adds to these immersive virtual design methods.

In Chapter 6 the software engineering aspect of the above work is discussed. Particular attention is paid to keeping the application fast and modular, supporting a number of different options depending on the task at hand.

Finally, Chapter 7 presents a review of the conclusions and suggestions for future work.

CHAPTER 2. INTERACTIVE MANIPULATION OF MESH-FREE MODELS IN VIRTUAL REALITY

For submission to: *The Journal of Mechanical Design*

Andrew Fischer, Judy M. Vance

1. Introduction

The mechanical design process involves both the creation of new products and the refinement of existing ones to produce an optimum design. Traditionally, designers create an initial product to solve a given problem and then analyze the stresses, strains, deformations, etc. the part will experience while in use. If these parameters exceed certain limits, the part is redesigned. An analysis is again performed, and the cycle continues until a satisfactory design exists.

This evolutionary path to improving existing designs is frequently encountered by engineers in practical problems. Thus, a large quantity of research focuses on refining this improvement process with various mathematical and automated optimization techniques. In contrast the less frequently encountered, but highly critical, initial creative design process has seen much less attention and formal computational aid [1].

This work seeks to develop a methodology that aids this initial design process and the early stages of design, before a working solution exists. The goal is to couple computer models with analysis models while allowing shape and design changes to be performed in real-time within a three-dimensional virtual environment. For purposes of this document, real-time will be considered as roughly less than $1/15^{th}$ of a second, a time increment below which most people are generally not able to visually distinguish discrete events, making for a visually smooth simulation. Providing this type of combined design and analysis environment with a minimum of restrictions should encourage the quick investigation of many possible shape and design changes and how they affect the final product performance and operation. Obtaining a better initial design will also aid the evolutionary product improvement process, making it faster and more straightforward.

A key component of these methods is the integration with virtual reality (VR). Working within a virtual environment provides the support for truly interactive design, where users can alter designs quickly while interpreting complex analysis results and evaluating the effects on a part or assembly of parts. A collaborative virtual environment allows groups of engineering designers and analysts to work together on fast interactive investigations of multiple part shapes early in the product design process. The more quickly and accurately analysis results can be approximated, the more options may be explored in a given time. By providing designers with quick and natural model interaction, the effects of shape and parameter changes on a part or assembly may be more easily evaluated.

At the Iowa State University Virtual Reality Application's Center, a C++ program, referred to as the Interactive Virtual Design Application (IVDA), has been created to implement and test these methods. It combines computer aided design (CAD) geometry, engineering analysis results, real-time shape manipulation with fast analysis approximations, and haptic force feedback in a virtual environment.

Another key to the methodology developed here is the use of mesh-free analysis methods in place of the traditional finite element methods commonly used in most engineering analysis software. Since large shape changes are expected from a designer working at the initial conceptual design stage, the analysis must be able to cope with these large changes without greatly degrading the results, and still be fast enough to approximate at real or semi real-time rates. A custom mesh-free method was implemented in the virtual reality program, based on a Reproducing Kernel Particle Method with Stabilized Conforming Nodal Integration for strain smoothing developed by Chen et al. [2].

However, one of the main challenges in working with a virtual environment is taking full advantage of the freedom offered to a designer. Virtual reality represents a different paradigm than the traditional keyboard, mouse and monitor computer interface. Correspondingly, different interaction methods need to be used to avoid limiting the designer. This paper explores the interaction techniques implemented in the IVDA and some of the benefits they provide.

This paper presents the background and research leading to the development of this program. Limitations of the existing application are then explored, followed by the specific techniques used to improve interaction with the virtual environment and their effectiveness.

2. Background on interactive design in virtual reality

The basis for this research began with Yeh and Vance, who presented a method to perform interactive stress analysis in virtual reality. They used linear Taylor series approximations based on pre-computed stress sensitivities and a rectangular Non-Uniform Rational B-Spline (NURBS) bounding volume to deform the part shape [3].

Using the linear Taylor series approximations require stress values (σ) and the first derivatives of these stresses with respect to some design variable (h), in this case the object shape. This derivative is referred to as the stress sensitivity for shape design variables. Using this information, and the change in the design variable (h), one can interpolate new stress values (σ') using equation 1 as shown.

$$\sigma' = \sigma + \frac{d\sigma}{dh}\Delta h \qquad \textbf{eq. 1}$$

This process, however, is limited by the low accuracy of the Taylor series stress approximations for large design changes and the need to perform a stress and sensitivity analysis before the virtual reality interaction could begin. The method also required the design area and bounding volume to be identified beforehand, limiting the freedom to interactively explore alternate designs. Figures 2.11 and 2.2 below shows a connection rod with its thickness being altered by this method while the stress levels change accordingly.

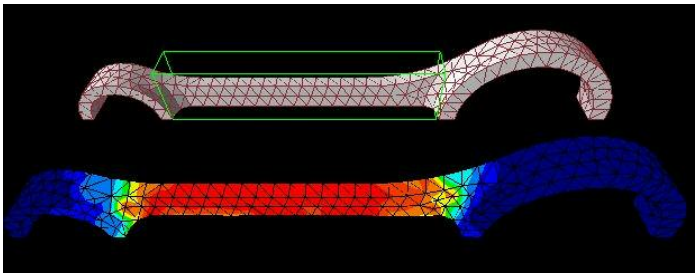

Figure 2.1. Initial configuration of a connecting rod with bonding volume visible.

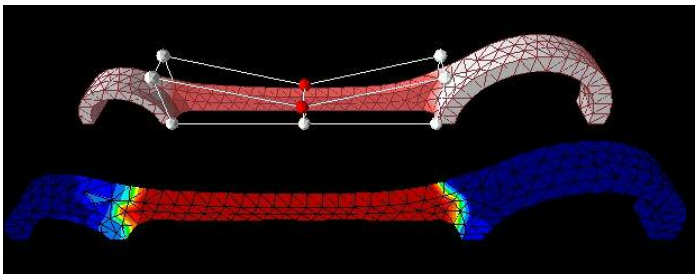

Figure 2.2. Deformed connecting rod with Taylor series stress approximations.

In the next stage, Ryken and Vance used a projection screen virtual environment to apply these techniques to a practical engineering problem, the design of a tractor rear lift arm [4]. This particular part experienced excessively high stress levels while in use, but it was difficult to alter its shape without interfering with the rest of the complicated lift assembly. Using the immersive virtual environment with real time stress approximations made it easy to explore the arm design and find a shape that decreased stress concentrations to acceptable levels while avoiding interference with the assembly. However, the method still required the user to determine the "changeable" area and bounding volume before the application was started, greatly limiting its flexibility. Figure 2.3 shows the lift arm in VR.

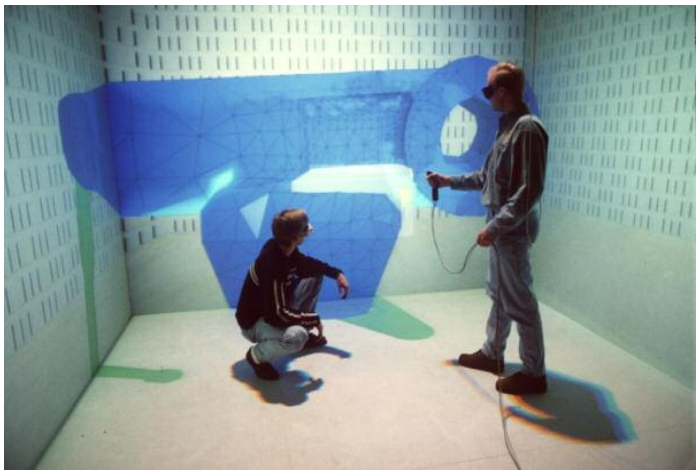

Figure 2.3. A tractor lift arm in virtual reality.

To further improve the immersive design process, Chipperfield, Yeh and Vance made two important additions. First, a reproducing kernel mesh-free method with strain smoothing stabilization was implemented to compute the analysis results. Second, a pre-conditioned conjugate gradient (PCG) re-analysis was added in addition to the existing Taylor series approximation to rapidly and accurately compute the stress contours resulting from shape changes [5].

2.1. Mesh-free analysis method

Generally, large changes in part shape and size cause traditional finite element methods (FEM) to suffer greatly from mesh distortion errors unless the mesh is reconstructed. Since reanalysis speed after part deformation is very important in this research, which attempts to make the interactive design as close to real-time as possible, it is very desirable to avoid the computationally expensive re-meshing process. This lead to the selection of mesh-free methods to solve for analysis results, as large geometry changes are an important goal in this research. The selected implementation was a reproducing kernel particle method with strain smoothing stabilization, introduced by Chen et al [2].

Mesh-free methods are in many ways similar to traditional FEM, except that the displacement approximations are given entirely in terms of the mesh-free nodes, so no element meshes are necessary [6]. This avoids the mesh distortion issue altogether. The

reproducing kernel is used to approximate unknown displacements in terms of the displacement coefficients at the mesh-free nodes. This relationship is shown in equation 2, where $u^h(x)$ is the displacement, $\Psi_I(x)$ is the reproducing kernel shape function evaluated at the point x, with respect to the I^{th} node, and d_I are the displacement coefficients.

$$u^h(\mathbf{x}) = \sum_{I=1}^{N} \Psi_I(\mathbf{x})\ d_I \qquad \textbf{eq. 2}$$

These kernel functions are not interpolating, so the displacement at a nodal point is not equal to the value of the displacement coefficient at that node. This makes displacement boundary conditions more difficult to apply in mesh-free methods.

As mentioned, the implementation used here is a reproducing kernel particle method (RKPM) with stabilized conforming nodal integration (SCNI). This SCNI acts as a strain smoothing approximation to eliminate spurious strains that arise from direct nodal integration of the RKPM equations.

If we let the strain at node x_L in terms of the displacement coefficients be given by equation 3 where $\boldsymbol{\varepsilon}^h(x_L)$ is the strain at node L, we then need: $\mathbf{G}_L$, a group of nodes that have shape functions with support covering the nodal region of node L; $\mathbf{B}_I$, the smoothed strain gradient matrix; and d_I, the vector of displacement coefficients for node I.

$$\boldsymbol{\varepsilon}^h(\mathbf{x}_L) = \sum_{I \in G_L} \mathbf{B}_I(\mathbf{x}_L)\mathbf{d}_I \qquad \textbf{eq. 3}$$

The smoothed strain gradient matrix is then given by the following three equations:

$$\mathbf{B}_I(\mathbf{x}_L) = \begin{bmatrix} b_{I1}(\mathbf{x}_L) & 0 \\ 0 & b_{I2}(\mathbf{x}_L) \\ b_{I2}(\mathbf{x}_L) & b_{I1}(\mathbf{x}_L) \end{bmatrix} \qquad \textbf{eq. 4}$$

$$b_{I1}(\mathbf{x}_L) = \frac{1}{A_L} \int_{\Gamma_L} \Psi_I(\mathbf{x})\, n_1(\mathbf{x})\, d\Gamma_L \qquad \textbf{eq. 5}$$

$$b_{I2}(\mathbf{x}_L) = \frac{1}{A_L} \int_{\Gamma_L} \Psi_I(\mathbf{x})\, n_2(\mathbf{x})\, d\Gamma_L \qquad \textbf{eq. 6}$$

Here A_L is the area of the nodal region (Γ_L) at node L and $n(x)$ is the normal vector to the boundary of the nodal region. The smoothed gradient matrix for node L is composed of integrals around the nodal region of node L multiplied by the area of the nodal region at node L, denoted A_L. Using SCNI requires one of these nodal regions to be associated with each mesh-free node in the geometry. In this work the problem of obtaining these regions was solved by using an already constructed FEM mesh as the basis for the mesh-free analysis, which also serves as a convenient way to import models into an application.

An initial finite element model, generated from an external analysis package, is used to define the geometry and to place the mesh-free nodes. By placing one mesh-free node per element, the existing element boundaries may be used as nodal regions for the SCNI. Since the mesh-free method doesn't require uniform nodal spacing for accuracy, regions on the edge of a model have the node placed on the region edge, while those within the model have the node placed in the middle. An example mesh-free node placement scheme appears in figure 2.4, where the cells indicate the original FEM geometry and the visible nodes are for the mesh-free analysis.

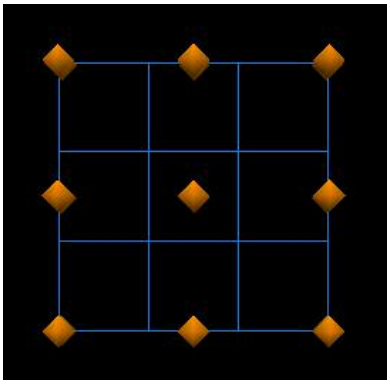

Figure 2.4. Mesh-free node placement in FEM model

2.2. Iterative PCG re-analysis

In order to make these interactive design methods truly practical they must be fast, reasonably accurate, and permit an engineer full freedom to explore design possibilities. This becomes even more important when trying to work in virtual reality, as the effectiveness of an application depends on real-time interaction [7].

Since analyzing a model is the slowest portion of this process, Chipperfield et al. implemented a pre-conditioned conjugate gradient (PCG) re-analysis to rapidly and accurately compute the stress contours resulting from design changes [8].

Solving the mesh-free equations require solving a linear system analogous to the finite element method:

$$\mathbf{K}d = f \qquad \textbf{eq. 7}$$

Here **K** is the sparse, square stiffness matrix, d is the vector of displacement coefficients, and f is the right hand side force vector. Once the design has been altered by deforming the geometry, a new system is formed:

$$\mathbf{K}^* d^* = f^* \qquad \textbf{eq. 8}$$

When using the PCG method for a reanalysis, the system has already been solved once for the full analysis. Hence the solution to the initial design is known ($\mathbf{K}^{-1}$). If the

design hasn't been drastically changed, the **K*** matrix for the modified design is similar to the initial **K** matrix. Because of this similarity, $\mathbf{K}^{-1}$ makes an excellent pre-conditioner to solve equation 8 with the PCG method. This technique seems to converge quickly, even for relatively large design changes.

This results in a two-stage process to compute stress contours while a model is being deformed in virtual reality, once the initial analysis is complete. Taylor series approximations are used for fast real-time stress updates while the more accurate, but slower, preconditioned conjugate-gradient re-analysis calculates stress levels when accuracy is important.

The goal is to balance the tradeoffs between real-time analysis results with a reasonably accurate solution necessary for design. Stress sensitivities are computed with a simple finite difference method that perturbs the selected portion of the model and resolves the problem. Though it lacks accuracy beyond small geometry changes, the Taylor series is fast enough to update the stress pattern in real time, allowing the designer to see the contour change as the model shape is altered. When more accurate results are needed, or a user wishes to modify other portions of the model, the pre-conditioned conjugate gradient re-analysis is performed on the model. This method computes a more accurate stress pattern and provides new stress sensitivities for the Taylor approximation.

2.3 Limitations

The application as described above provides mesh-free analysis of single engineering parts with fast Taylor Series approximation for stress pattern changes, free-form deformation, and an iterative Preconditioned Conjugate Gradient method for more accurate, albeit more computationally expensive, reanalyses. However, while it demonstrates several powerful concepts, certain limitations prevent the program from being a practical design tool.

To allow the designer as much freedom as possible in the virtual environment, he/she should be able to rapidly shape change different portions of the model geometry. In the work presented above, the bounding volumes used for free-form deformation must also be defined with external software, or by hand in an input file, and loaded into the

virtual environment. This is unacceptable, as the designer needs maximum freedom to explore different design shapes.

The engineering design process is rarely concerned with a single model by itself. Typically, parts need to be seen in the context of their surrounding geometry, which can influence and interfere with the changing design. To meet this requirement multiple analysis models with collision detection and methods to prevent geometry interpenetration are needed.

Finally, quickly understanding the analysis state of a model, let alone several models in the design space, is a challenging task with all of the information available for the designer to consider. It makes sense to explore additional channels of information feedback available to the user in the virtual environment. One possibility is the use of force or haptic feedback to allow the user to "feel" the changing stress state of a model.

The above limitations are considered and improved upon in the following section, along with an update outline of the IVDA program.

3. Interactive design methods

As stated, the primary goal of this work is to provide a designer with the maximum freedom to rapidly modify the analysis geometry in a virtual environment and interpret the results. The sections below detail the most significant ways this interaction is achieved.

3.1 Working in Virtual Reality

The IVDA application is brought into virtual reality using VR Juggler, an open source software library developed at Iowa State University Virtual Reality Applications Center. VR Juggler provides the framework for programs to run on a wide variety of virtual reality devices, from multi-screen projection environments to simple head mounted displays; or even in simulation mode on a standard desktop computer [9].

These methods were designed to be applied in a large projection screen environment, since it provides a user the most freedom to explore and alter the geometry and visualize results. Examples of such systems include the C6 at Iowa State University

[10], and commercially available systems such as the FLEX from Fakespace systems [11]. Interaction is achieved via a tracked wand, which the designer uses to make decisions and manipulate geometry. An interactive menu system was implemented, permitting the user to rapidly move between the different program options. Since the menus are a full 3d object, they may be positioned with total freedom in the virtual environment.

Originally the exclusive realm of large multiprocessor graphics workstations, large projection screen virtual reality has recently begun making extensive use of lower cost alternatives, such as synchronized clusters of high-end PC's. The IVDA has followed this evolution, taking advantage of the opportunities for parallel processing across a networked cluster. The mesh-free solver, originally written using SGI specific numeric routines for calculation, was re-factored to work with the cross-platform SuperLU software [12]. SuperLU provides the direct solution of large, sparse systems of linear equations on high performance computers. The SuperLU_DIST version of the software permits parallel processing on distributed memory parallel computers, just like those found in a VR cluster.

3.2 Free-form deformation

To alter part geometry in the virtual environment, a form of subdivision volume free form deformation (FFD) is used in the application. Free form deformation is based on Non-Uniform Rational B-Spline or NURBS modeling [13]. Here, Catmull-Clark subdivision volumes are applied as an extension to free-form deformation techniques. This allows more variety and flexibility in defining the control volume around the part where shape deformations are allowed than standard NURBS-based free form deformation [14].

Using this type of free form deformation provides a useful way to manipulate geometry in the virtual environment. A volume is defined around the portion of the geometry to be changed by placing a series of control points in the 3D space surrounding the part. Once complete, a series of subdivisions are performed on the control point volumes to produce a smoother shape. The part geometry is mapped to the subdivided

volume, allowing the shape to be changed by manipulating the different control points. Designers in VR need only grab and move these controls points making for a natural interaction method.

However, defining the initial bounding volume or volumes while in VR is not a trivial task. A number of different options were considered to replace loading the volume from a file, including:

1. Placing the volume points one by one.
2. Placing a single volume and extruding surfaces to create more
3. Dropping in bounding boxes individually and linking them together
4. Using the wand to "rubberband" a box around the portion of the model and subdividing it into smaller boxes

These different techniques were implemented and tested in the application for usability. A summary of the findings are as follows:

Placing the points one by one is a tedious operation in the virtual world. Building a single bounding box requires 8 different point selections, and at least 4 more for each additional box. It is also difficult to tell what portions of the model are being covered until the volume is complete, so point positions must usually be improved individually after the fact.

Extruding a single volume is easier than placing individual points. The volume may be positioned and scaled as desired. Repeatedly extruding surfaces to produce several control points for fine deformation control, however, still takes plenty of time and often results in uneven volumes that require additional point movement. If the user wants to change the control point density in a certain area, he/she is required to manually shift a series of bounding box planes about.

Linking bounding volumes together to cover the areas of interest is faster than extruding, but still becomes a challenge when the user wants multiple control points in all 3 coordinate directions. A large number of boxes need to be linked, and it can become difficult to tell what kind of overall volume has been built, as the appearance is quite confusing.

Rubberbanding a single bounding box around the whole area of interest is the fastest way by far. The user simply selects two points in space. A changing volume

follows the wand once the first point is selected (and the wand orientation noted for volume orientation) until a second is placed. To change the control point density, the volume may be repeatedly subdivided in any of the box's local coordinate directions.

This makes it very easy to add a large number of control points for fine grained model deformation. Multiple rubberbanded volumes may be used on a single model as well, in case several areas of interest arise. Note that this technique's subdivisions are distinct from the subdivision surface operations used to embed the model in the bounding volumes.

The rubberband/subdivide technique was selected for use in the IVDA after testing indicated it's simplicity and speed advantages. Subdivided bounding volumes placed around a simple model appear in Figure 2.5 below.

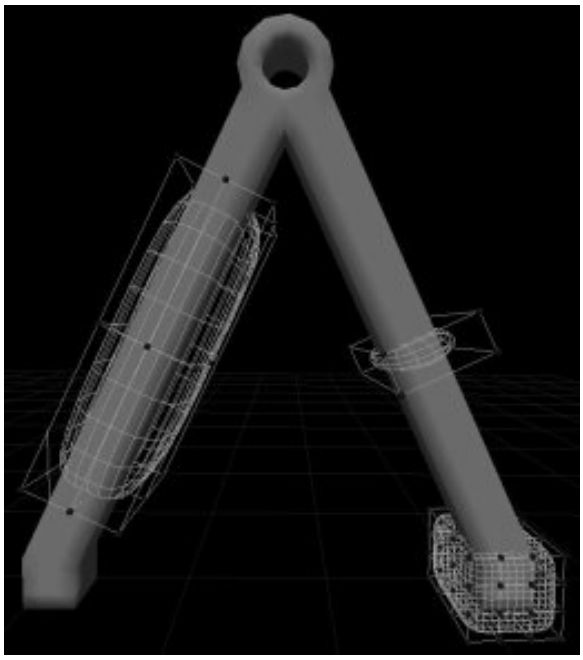

Figure 2.5. Subdivided bounding volumes around a model in the virtual environment

3.3. Collision detection

Since the advantages of doing interactive design are even more apparent when working with a large assembly of models in VR, detecting and handling collisions between the many models is critical. Along with the more typical uses for collision

detection, in the IVDA the designer should be prevented from changing a model's geometry to pass through another part in the assembly. Selecting a collision detection technique was guided by some requirements that are specific to interactive design.

Speed is of primary importance to keep the application in real-time. While most collision detection problems are concerned with many solid or rigid bodies colliding with one another, this research is primarily concerned with collisions between deforming models as they change shape, to avoid passing though other geometry. Such deforming model collisions should be checked for without rebuilding the entire collision detection structure as this is typically an expensive computation. Finally, since the mesh-free analysis of large models can use a substantial portion of the available computer memory, efficient use of memory by the collision detection routines is another important issue.

Working within these constraints led to the selection of the OPCODE collision detection library (OPtimized COllision DEtection) for this research. With routines based on the well-known RAPID package, OPCODE was authored by Pierre Terdiman to be a more memory efficient implementation [15]. OPCODE has proved popular in computer game development with physics engines such as the Open Dynamics Engine (ODE), where speed and efficient memory use are critical [16].

OPCODE provides fast collision detection between non-convex polygonal models with a memory footprint many times smaller than similar packages such as SOLID and RAPID [17]. OPCODE has recently added support for colliding deforming polygonal models without rebuilding the entire collision detection structure, making it a natural choice for this work, since deforming model collision queries are so important.

Integration was provided by creating a translator to convert the application's analysis model to an OPCODE collision model. Since the original geometric model is a finite element mesh, and OPCODE performs collision queries on a per-triangle basis, the conversion is straightforward. Pointers to the original model's nodal coordinates are maintained, so as the shape is altered by interactive design the OPCODE model updates accordingly.

Collisions are reported back to the designer by preventing further model movement or deformation. Since OPCODE reports the collision triangles and normal

direction, it is also possible to indicate the specific collision location, useful for complicated models or assemblies.

3.4. Haptic feedback

Haptics, or force feedback, refers to the application of touch sensation and control to human-computer interaction applications, allowing users an additional channel of information in a simulation. In the framework of this research project, haptics is being used as an experimental tool to provide designers with information about the stress state of a part as it is deformed in virtual realty. A PHANTOM haptic device, originally developed as a low-cost desktop tool, is used to generate this force feedback [18].

Since the PHANTOM is primarily a desktop device with a physical workspace much smaller than the interior of most virtual environments, a mapping was developed to use the phantom over an arbitrarily large portion of the virtual space [19]. The user simply defines a workspace bounding volume around the virtual environment area of interest while an algorithm maps the PHANTOM physical motion to the virtual space, with a virtual PHANTOM endpoint displaying the PHANTOM location. This permits the user to work in areas ranging from a small section of the environment (even smaller than the haptic device workspace) to the entire virtual world.

Force feedback is sent to the user when the virtual endpoint is used to manipulate a control point of the FFD volume, deforming the model and changing the stress pattern. By altering the force required to manipulate the control point based on a weighting of the stresses in the model, the user receives additional information about how the stress levels are affected by part shape changes that may not immediately be visually noticeable. Haptics, combined with collision detection routines, can also be used to prevent the user from deforming the part further if another model is struck or a certain value of stress is exceeded. Haptic may even replace the wand for most application interaction, such a point selection or moving models about the environment.

Since haptic control loops require high update rates (~1000 Hz) for stability, Kim and Vance presented a method where the haptics and PHANTOM control are executed on a separate haptic computer, which is networked to the computer(s) driving the VR

simulation [20]. The haptic computer then maintains a complete and separate simulation from that of the VR system, and the necessary data is exchanged to keep them both up to date. Keeping the haptic process separate and communicating via TCP ensures the necessary high update rates, and it allows the PHANTOM setup to be more portable, as it only requires a network connection.

This design is particularly well suited to the interactive design application, since only a small amount of data actually needs to be transferred between simulations once they have both been started, specifically the location of the PHANTOM endpoint and the force value based on the stress patterns. Maintaining haptic update rates are thus not a problem. Figure 2.6 shows a PHANTOM haptic device in the C6 virtual environment.

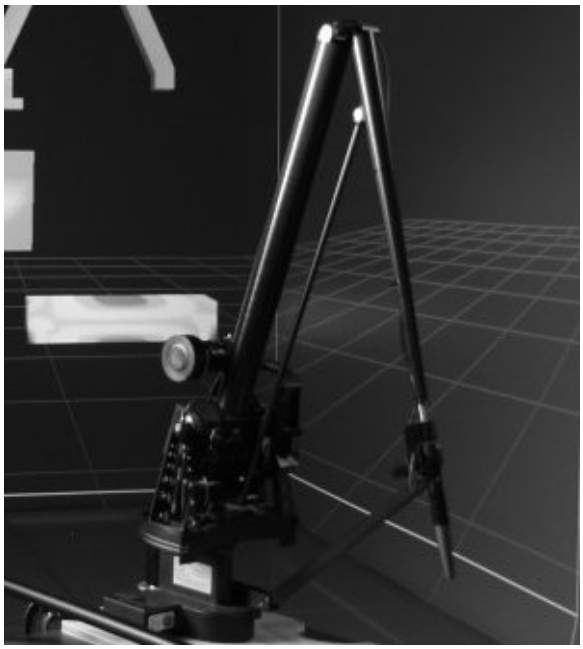

Figure 2.6. A PHANTOM inside the C6.

4. Application sample use

An intuitive environment for observing the effect of shape changes on stress distributions has been presented in its current state. Using the program is straightforward, as per the following example.

Model information, boundary conditions, and material properties are stored in an XML file. This file is created from an ABAQUS input file by the program or by a user. Once the program is running, a model is selected from those available and loaded into the VR environment. The application performs an initial stress analysis and sensitivity calculation for selected control points. A user then defines a bounding volume around the part as the allowable deformation area. From there a designer can use the wand or the PHANTOM to modify the model, select different control points, change bounding volumes, and explore the shape change effects on the part in question.

The application has the option to display different contours based on Von Mises stress, maximum shear, etc. as in a standard finite element software package. Multiple models may be loaded into the environment, properly positioned, and each analyzed in turn. Collision detection and haptic feedback may be turned on or off as needed. Figure 2.7 provides a diagram of typical program operation.

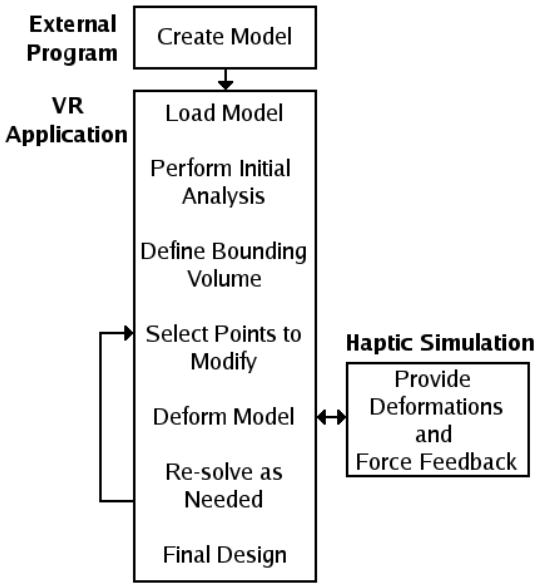

Figure 2.7. Flowchart for program operation

5. Conclusions

The application as described works well with multiple models, large shape changes and linear elastic analysis. Improvements to the interaction methods make it easy

for a designer to quickly alter a number of different models and immediately observe the effects, all within a virtual environment. The addition of haptic feedback provides a novel interaction method worthy of additional research.

6. Acknowledgements

The authors would like the thank the Virtual Reality Applications Center for the use of computations resources and hardware.

7. References

1. Kim, N.H., *Design Optimization*, in *The CRC Handbook of Mechanical Engineering*. 2004.
2. Chen, J.S., et al., *A stabilized conforming notal integration for Galerkin mesh-free methods.* International Journal for Numerical Methods in Engineering, 2001. **50**: p. 435-466.
3. Yeh, T.-P. and J. Vance, *Applying Virtual Reality Techniques to Sensitivity-Based Structural Shape Design.* ASME Journal of Mechanical Design, 1998. **120**(4): p. 612-619.
4. Ryken, M.J. and J.M. Vance, *Applying Virtual Reality Techniques to the Interactive Stress Analysis of a Tractor Lift Arm.* Finite Elements in Analysis and Design, 2000. **35**: p. 141-155.
5. Chipperfield, K., T.-P. Yeh, and J.M. Vance. *Interactive product development in a virtual environment utilizing haptics*. in *2002 NSF Design, Service and Manufacturing Grantess and Research Conference Proceedings*. 2002. San Juan, Puerto Rico.
6. Hardee, E., et al., *A Structural Nonlinear Analysis Workspace (SNAW) based on meshless methods.* Advances in Engineering Software, 1999. **30**: p. 153-175.
7. Jayaram, S., et al., *Assessment of VR Technology and its Applications to Engineering Problems.* Journal of Computing and Information Science in Engineering, 2001. **1**: p. 72-83.

8. Chipperfield, K., J.M. Vance, and A. Fischer, *Fast Meshless Reanalysis Using Combined Approximations, Pre-Conditioned Conjugate Gradient, and Taylor Series.* AIAA Journal, 2006. **44**(6): p. 1325-1331.
9. Bierbaum, A., et al. *VR Juggler: A Virtual Platform for Virtual Reality Application Development*. in *Virtual Reality, 2001*. 2001. Yokohama, Japan: IEEE.
10. Virtual Reality Applications Center, *About the C6*. VRAC Intranet. 2001. https://intranet.vrac.iastate.edu/vr/c6/info/page.html. July 16, 2006.
11. Fakespace Systems, *FLEX and reFLE.,* Fakespace website. 2006. http://www.fakespace.com/flexReflex.htm. July 20, 2006.
12. Demmel, J.W., J.R. Gilbert, and X.S. Li, *SuperLU Users' Guide*. 2003. Included with SuperLU software.
13. Piegl, L. and W. Tiller, *The NURBS Book*. 1997, Berlin: Springer.
14. MacKracken and Joy. *Free-Form Deformations With Lattices of Arbitrary Topology*. in *SIGGRAPH 96*. 1996.
15. Terdiman, P., *Memory-optimized bounding volume hierarchies*. 2001. Included with OPCODE software.
16. Smith, R., *Open Dynamics Engine v0.5 User Guide*. 2004: Included with ODE software.
17. Terdiman, P., *OPCODE User Manual*. 2002. Included with OPCODE software.
18. Massie, T. and J.K. Salisbury. *The PHANTOM haptic Interface: A Device for Probing Virtual Objects*. in *ASME Symposium on Haptic Interfaces for Virtual Environments*. 1994. Chicago, IL: ASME.
19. Fischer, A. and J.M. Vance. *PHANToM Haptic Device Implemented in a Projection Screen Virtual Environment*. in *7th International Immersive Projection Technologies Workshop*. 2003. Zurich, Switzerland.
20. Kim, C. and J.M. Vance. *Development of a networked haptic environment in VR to facilitate collaborative design using Voxmap Pointshell (VPS) software*. in *ASME Design Engineering Technical Conferences and Computer and Information in Engineering Conference*. 2004. Salt Lake City, Utah.

CHAPTER 3. FAST MESH-FREE REANALYSIS WITH OPEN SOURCE SOFTWARE FOR VIRTUAL DESIGN

For submission to: *AIAA*

Andrew Fischer, Judy M. Vance

1. Introduction

The evolutionary process of mechanical design typically requires many iterations on an initial design achieve an acceptable or optimal solution. Modern advances in computer aided design, analysis, and optimization have led to a substantial amount of research in speeding and automating this process. However, the early stages of engineering design, where no initial design of any form exists, has seen far less attention and rigorous computational aid [1].

At the Iowa State University Virtual Reality Application's Center, an Immersive Virtual Design Application (IVDA) has been created to develop methods for aiding the initial design process. Computer aided design (CAD) geometry, engineering analysis results, real-time shape manipulation, fast analysis approximations, and force feedback are combined into a virtual workspace for designers. Working a virtual reality provides a truly interactive design space, where geometry may be quickly altered and complex analysis results interpreted. Collaboration is another important benefit of VR, since it lets groups of engineering designers and analysts to work together on the same products at the same time.

In this research, mesh-free analysis methods are used in place of more traditional finite elements to help deal with the large shape design changes expected from a designer working within the freedom of a virtual environment. A custom mesh-free method was implemented in the IVDA, based on a Reproducing Kernel Particle Method with Stabilized Conforming Nodal Integration for strain smoothing developed by Chen et al. [2].

However, while suitable as a proof of concept tool, implementing a custom mesh-free analysis makes it very difficult to deal with the wide variety of engineering problems and analysis types that could be encountered by a designer, due to the sheer number of different analysis options that must be available. A more suitable alternative would be to combine the aforementioned virtual reality application with some external analysis software that supports mesh-free analyses and blends well with the existing program.

An intriguing option for the external analysis is the use of Open Source Software (OSS). Open Source refers to any software released under a license conforming to the Open Source Definition (OSD). Such a license must meet the following conditions:

1. The source code is available
2. The software is re-distributable
3. The software must be modifiable
4. The license must not discriminate against any users or field
5. The license must apply to all parties with the software
6. The license cannot restrict aggregations of software

The last 10 years have seen a significant growth of Open Source software as a viable alternative to commercial products in many areas, since OSS has been shown to quickly and inexpensively produce high quality software [3]. High profile examples include the Linux kernel and the Apache web server.

The remainder of this paper will cover a review of potential analysis software, both commercial and open source, for use with the application. The selection process is detailed, and the implementation and testing of the Tahoe analysis package with the current application is presented [4].

2. Software review

The application described above works well when applied to large shape changes with linear elastic analysis. However, relying on the linear elastic case presents a serious limitation when attempting to solve real world engineering problems, where material models and loading conditions require a more advanced analysis. Integrating external

analysis software with the existing application could provide a highly practical solution, if certain conditions are met.

To determine the best analysis software to integrate with our program several criterions were selected. The package that best met these criterions would then be selected for implementation and testing against our current analysis. These criterions include:

- Support for mesh-free analysis methods. While these interactive design methods have used traditional finite elements, the use of mesh-free methods with large geometry changes has proven beneficial in the project thus far, and dropping them would make the mesh distortion/re-meshing a problem.
- Support PCG reanalysis. While many software packages support the PCG method as a nonlinear analysis option, using it for our reanalysis method has some specific requirements [5]. Providing direct access to the matrix equations in the solver would permit great flexibility here.
- Provide fast access to data, both input to and output from the analysis process. This is important since a potentially large amount of data needs to be shared between the VR program and the analysis software. File I/O should be avoided, especially for the reanalysis process, since it can be large performance bottleneck.
- Support a wide variety of material models and analysis types.
- Well documented

A number of different analysis software was examined as a potential alternative to our current methods. A brief overview of several investigated packages follows. The packages are, for convenience, divided into commercial offerings and Open Source alternatives.

2.1. Commercial analysis software

These commercial packages were investigated to compare them with each other for use with our interactive design methods. As commercial offerings, they are assumed to be robust, well documented, and supplied with a large number of material models and

solver options. The focus is on whether they provide any mesh-free analysis methods and what kind of data access/scripting interface they provide.

ABAQUS, by ABAQUS, Inc, is an advanced finite element analysis software package. It supports a wide range of material models, element types, and analysis options [6]. It is also currently used to generate input files for the interactive design program, so its familiarity would make it easier to integrate with our application. ABAQUS provides a python scripting language for creating models, submitting jobs, writing input files, and obtaining output. A C++ interface exists as well, though it only permits I/O with the output database. In the version examined (6.5), however, ABAQUS does not provide support for mesh-free analysis methods.

ANSYS, Inc's finite element software boasts many elements, material models, and "the most comprehensive set of solvers" [7]. A scripting interface exists using the Tcl/Tk language. However its main use appears to be GUI development and extension. Access to program analysis data is more limited. Mesh-free methods also do not appear to be an option with any module.

LS-DYNA is provided by the Livermore Software Technology Corporation (LSTC), and is a "general purpose transient dynamic finite element analysis program" [8]. LS-DYNA has many elements, materials, and solvers; plus it runs on a large number of different hardware platforms and operating systems. In addition, recent versions support mesh-free analysis methods [9]. Little information, however, is provided on scripting languages or other interfaces.

FieldMagic is provided by Intact Solutions, LLC., and is a commercial offering based on the SAGE software developed by Shapiro, et al. [10]. It is interesting in that is it a completely mesh-free analysis tool. FieldMagic supports thermal, vibrational, and structural analysis modes [11]. However, it is currently a two-dimensional tool, so it isn't as useful given the three-dimensional nature of the interactive design process.

Overall both ABAQUS and LS-DYNA are attractive options. ABAQUS due to the flexibility provided by its scripting interfaces and the author's existing familiarity, and LS-DYNA due to the presence of mesh-free analysis methods. Still, each appears lacking in at least one area, and in general the choice of a commercial package places great

restrictions on the ability to add functionality to the software or directly access additional analysis data beyond what is provided. The advantages of this low level access make Open Source packages an attractive alternative.

2.2 Open source analysis software

Finding generally useful open source analysis packages can be difficult, due to the large number of outdated, partially finished, or highly specialized choices. However, since they provide direct access to the program's source, any information about the analysis is readily available making them easy to integrate with the interactive design application. A search of academic literature was made, as well as Internet compilations such as the open directory [12]. The focus was on finding software that is reasonable popular, actively developed, and supports some form of mesh-free analysis. A sample of the more relevant programs follows.

The Impact project is hosted on sourceforge.net, a popular repository of open source software, and provides a free explicit dynamic finite element program [13]. It is mainly designed for dynamic events involving large deformations. Impact supports three-dimensional analyses, and is an active project, though development has slowed recently. Impact does not provide any mesh-free methods, and is written in Java, while the interactive design application uses C++.

Calculix is a three-dimensional finite element program for structural analysis. It handles static, dynamic, and thermal analyses. Calculix also supports loading ABAQUS input files. Their site also showcases some impressive examples [14]. However, only a few developers seem to have contributed to the project. There is no mesh-free support, and it is difficult to determine how actively the project is developed or maintained.

Tahoe is a "research-oriented, open source platform for the development of numerical methods and material models" with a goal toward "the simulation of stresses and deformations for situations that cannot be treated by standard continuum simulation techniques" [15]. Tahoe provides support for mesh-free methods, including a Reproducing Kernel Particle Method similar to the one implemented in the interactive design application. Tahoe is quite actively developed, and provides extensive

documentation as well as a user's forum for questions. This documentation, combined with the program's clean structure, make the source code easy to understand and modify.

Among the open source packages surveyed, Tahoe was the clear leader. Its support for mesh-free methods for 3D problems and clean C++ design would make it easy to integrate with the interactive design application. And since it is actively developed and used, its methods are anticipated to be robust and accurate.

2.3 Selection

After analyzing each software package according the stated criterions, the decision was made to integrate Tahoe with the interactive design application. While both ANSYS and LS-DYNA are attractive options, if Tahoe provides sufficiently fast and robust methods for our application, it's ease of integration and open code would make it an excellent choice. Following the Open Source development model, any changes or improvements to Tahoe could be submitted back for implementation in the main project code base, benefiting both applications. The follow section details the integration of Tahoe with the existing interactive design application.

3. Integrating Tahoe

To implement Tahoe as a portion of the application, it is helpful to examine the structure of both programs. The interactive design application is created around a core class that uses VR Juggler to drive the virtual environment. Other program modules handle different tasks such as menu interaction within VR, file I/O, and free form deformation. Model geometry and mesh-free node information is stored in a manager, while the mesh-free solver uses the SuperLU software [16], to actually solve the matrix equation, and provides the PCG method for fast reanalysis. The Taylor series approximations to stresses are fairly trivial to compute once sensitivities have been obtained by the solver, so they are handled separately as the model is deformed. A simple diagram of the application appears in figure 3.1.

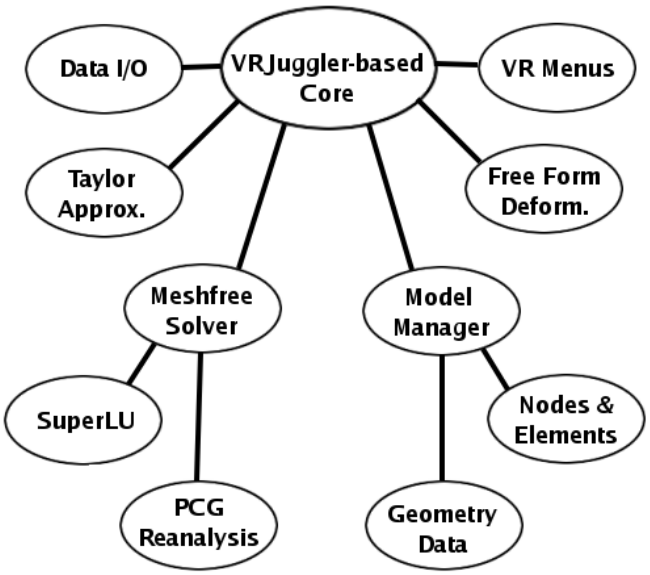

Figure 3.1. Interactive design application structure

The design of Tahoe is similar in some ways. The overall program is controlled by a tahoe-manager, which provides communication and exception checking. Individual managers control and manipulate different data, such as the elements in the element group, or the solver, which equilibrates system of matrix equations. Full details appear in the Tahoe User's Guide, which accompanies the software. Figure 3.2 demonstrates this structure, and shows some similarities to the interactive design program.

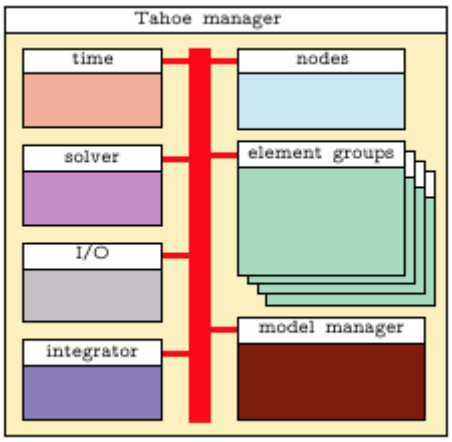

Figure 3.2. The structure of Tahoe
Courtesy of the "Tahoe Users Guide"

3.1 Implementation

There were two goals guiding this integration: 1. Make the use of Tahoe indistinguishable from the original mesh-free solver to the rest of the application. 2. Minimize any changes to the Tahoe software itself. Since Tahoe is an open source application, it was possible to examine the program code to determine the best integration method and use that code as a guide.

The first step was to create an interface between Tahoe and the interactive design application. This interface, called the tahoe solver, replaces the mesh-free solver module in figure 7, and provides the same interface methods to the application. The tahoe solver takes a model, including geometry, material properties, and boundary conditions, from the model manager and formats them in a structure Tahoe understands. Updated model shapes from the free form deformation are passed directly into the Tahoe model, as well as any changes to model properties. Information from the Tahoe analysis, such as the resulting stress tensors, is then returned to the application. This avoids the use of any file I/O during the reanalysis process.

Since Tahoe is designed to be command line driven, some additional work is needed to control it from the interactive design application's tahoe solver module. This was accomplished by extending the tahoe manager class to create a custom manager. The custom manager has all the functionality of the original tahoe manager, but it provides

additional accessor methods to the underlying Tahoe data, and receives commands directly from the solver module.

While it was desirable to avoid changing the Tahoe source code, there were some areas where it was necessary to access internal information. These changes were the addition of simple accessor methods to the relevant class files. The GNU "patch" command is used to apply these additions to Tahoe automatically, so they may be easily added to updated Tahoe versions.

3.2 Input and output

Typically a Tahoe analysis requires two files, a geometry file and an XML input file. The geometry file contains all the information for a finite element model such as nodes, elements, and groups of each. Input files list the parameters for the analysis such as element types, material properties, boundary conditions, and solver controls.

Since Tahoe is a complex analysis package with a wide variety of options, it becomes impractical to control all of these options from within the virtual environment, where interaction is limited to a multi-buttoned wand for simplicity and a menu system. However, it is also desirable to avoid forcing the user to create or edit this input file any time they wish to perform an analysis. As a compromise, the interactive design application builds a skeleton Tahoe input file for a mesh-free analysis from the limited options available to the user in the virtual environment. This skeleton file may be used as is by the tahoe solver, or edited by hand or with a graphical XML editor to further tune the analysis.

The Tahoe geometry file is created from the initial ABAQUS input file by the Data IO module. Tahoe provides some support for the loading of ABAQUS input files, but it is apparently only usable for those generated with ABAQUS 5.x; not the 6.5 version used here. This geometry file is only used for the full analysis at the start of the interactive design program. For subsequent reanalyses the Tahoe Manager fills in the deformed model geometry and properties directly.

3.3 Virtual Environment Issues

As mentioned, controlling a sophisticated analysis package like Tahoe from within a virtual environment presents some challenges. There are a wide number of input file options to control. To make this easier, the application makes it easy to specify which particular parameters should be changeable from within the virtual environment, and how those parameters affect filling in the skeleton input file. A special menu is used where the user may toggle on/off different options, or alter numeric values with a slider bar that lets the user move within a range.

This method of interaction is a good compromise between forcing the user to accept default solver values, changing them only when outside the virtual environment, and having the user modify the entire input file while in VR. That would be a time consuming task even with a hand-held tablet PC, and would force the user to carry another piece of equipment into the environment. By choosing parameters to be modified interactively, the user may be presented with only the most relevant options for the design they are modifying.

Once Tahoe was implemented as an optional solver in the interactive design application it becomes desirable to compare its analysis results to the homegrown mesh-free code originally used. While it is a given that Tahoe supports far more material models and analysis options, the authors wanted to see if it presented any advantages in speed and/or accuracy in the linear elastic case. The following section details this comparison.

4. Testing Tahoe

There are two goals in testing Tahoe's mesh-free methods with the current interactive design application's mesh-free implementation, referred to here as the “M3d” solver. The first is to compare the accuracy of the two packages with each other against a problem with a solution provided by commercial software using standard finite elements, in this case ABAQUS. The second is to compare the speed of analysis and reanalysis for each. For the full analysis this speed is less critical, since it is not something the user will do often in the virtual environment. The speed of reanalysis is much more critical, since

experience indicates a designer in VR only waits about 30 seconds for a reanalysis to complete before becoming distracted from the task at hand.

Though the PCG reanalyses is used in both cases, its solution is only part of the reanalysis process. The PCG method uses the deformed model stiffness matrix in its calculation. That means the time to compute this new stiffness matrix is quite important, even more so when one considers that mesh-free methods take longer to assemble this matrix than standard finite elements [17]. For smaller problems, the matrix assembly process often takes longer than the PCG solution step.

4.1 Example Problems

Three example problems were used in these tests. The first two consist of a beam with a uniform displacement on one end. The third is a brick with a circular hole under tension. Even though they may be solved with 2D assumptions, these problems were all done with a full 3D analysis. The limitations of the M3d mesh-free implementation limit the selection of example problems, since they must be linear elastic.

4.1.1 Example 1

The first example uses a 2cm x 2cm x 10cm beam with a uniform displacement of 0.001cm in the vertical direction applied to one end. The other end is fixed. Material properties used are Young's modulus: 200 GPa and Poisson's ratio: 0.3. Figure 3.3 below diagrams this example case in two dimensions. The displacement here has been exaggerated for clarity.

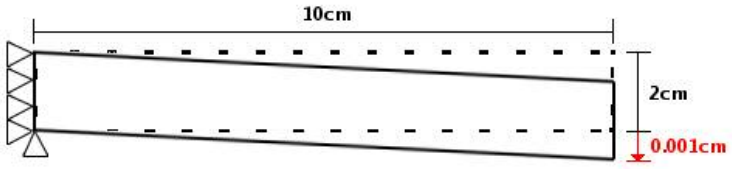

Figure 3.3. Beam with a uniform displacement

This example was modeled in ABAQUS with 625 elements and exported to an input file. The input file was first solved with ABAQUS, and then loaded by the interactive design application for use with the Tahoe and M3d analyses.

4.1.2 Example 2

The second example problem is analytically identical to the first. The only difference was the beam model is now discretized into 1617 elements. This tests to see if both methods give reasonable answers as the number of elements grows, and to compare timing data.

4.1.3 Example 3

The third example problem is a hole in a brick under tension due to displacement. Only a quarter model is used to take advantage of symmetry. This model is 10cm x 10cm and 5cm thick with a 2cm radius hole. Again material properties are Young's modulus: 200 GPa and Poisson's ratio: 0.3. The quarter brick model uses symmetry on the bottom and left edges, while the top edge has been displaced by 0.001cm in the upward direction. Figure 3.4 shows the quarter model.

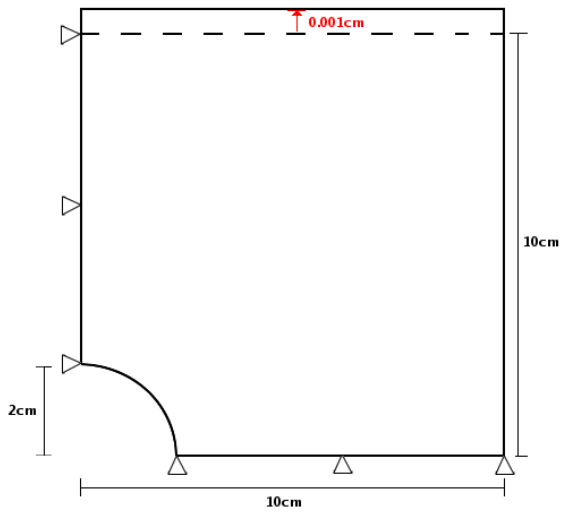

Figure 3.4. Quarter brick with a hole

This model was created in ABAQUS with 2574 elements, exported to an input file, and solved by each of the three methods.

4.2 Results

Each example problem was tested on a Red Hat Enterprise Linux workstation with dual 3.6Ghz processors and 3GB of RAM. To compare results Von-Mises stresses were obtained and plotted verses the analytical solution. To time each example problem, both the Tahoe and M3d analyses were run 10 times. The times for matrix assembly and system solution were recorded and averaged.

4.2.1 Example 1

In this example the Von-Mises stresses were computed by the application and taken from the top center row of the beam for comparison with a graph. They were plotted along with the ABAQUS solution in Figure 3.5. Timing data appears in Table 3.1.

The results show that both of the solvers give very similar stresses, and agree well with ABAQUS. Some unevenness does appear at the fixed edge of the beam, but this is not surprising given the coarseness of the mesh. The timings are also similar, though Tahoe does perform a faster assembly of the stiffness matrix.

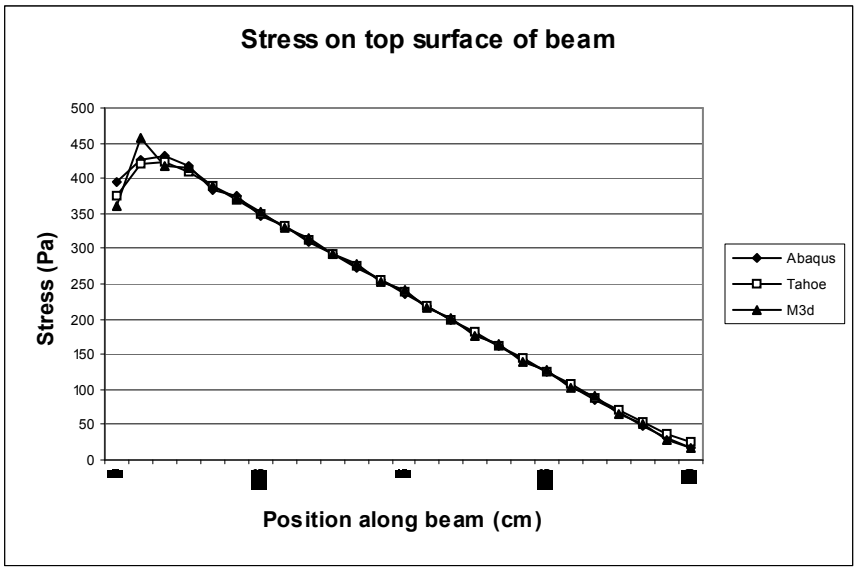

Figure 3.5. Von-Mises stresses along top center of beam

	Tahoe		***M3d***	
Run	Assembly	Solution	Assembly	Solution
1	0.18	2.8	0.85	2.97
2	0.18	2.82	0.85	2.9
3	0.18	2.81	0.86	2.91
4	0.18	2.81	0.86	2.91
5	0.18	2.8	0.85	2.77
6	0.18	2.81	0.85	2.91
Average	0.18	2.81	0.85	2.9

Table 3.1. Matrix assembly and solution times in seconds

4.2.2 Example 2

Similar to example 1, here the number of elements in the beam was increased. Results again show both solver stress plots and solution times largely in agreement; see figure 3.6. The timings, however, demonstrate Tahoe's clear advantage in the matrix assembly step. It was on average almost six times faster as shown in Table 3.2.

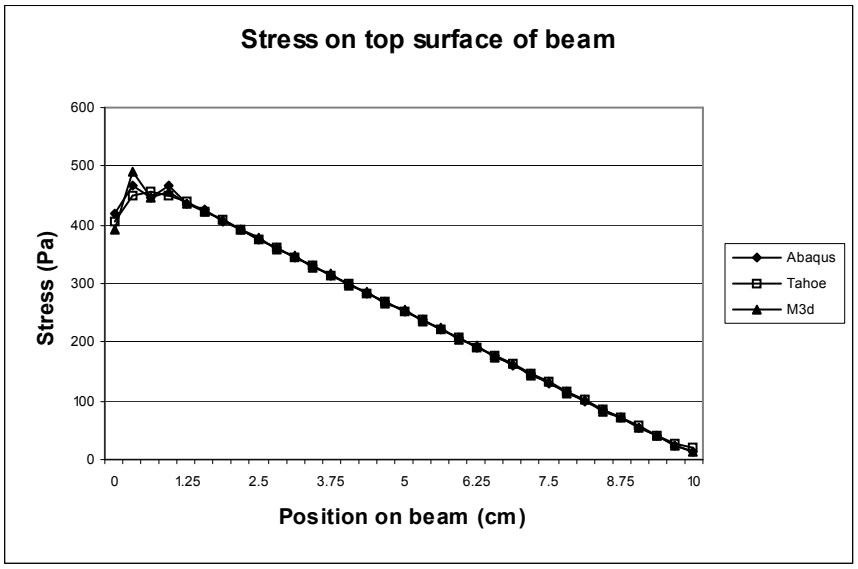

Figure 3.6. Von-Mises stresses along top center of beam

	Tahoe		***M3d***	
Run	Assembly	Solution	Assembly	Solution
1	0.55	13.54	2.89	14.94
2	0.55	13.48	3.05	15.39
3	0.55	13.51	2.51	11.83
4	0.55	10.9	3.1	15.39
5	0.57	11.69	3	14.84
6	0.53	13.56	3.03	15.3
Average	0.55	12.78	2.93	14.61

Table 3.2. Matrix assembly and solution times in seconds

4.2.3 Example 3

The last example uses Von-Mises stresses along the bottom center of the brick, starting at the edge of the hole and moving away. As with the first two examples, the stresses between Tahoe, M3d, and ABAQUS are almost indistinguishable. This example also lacks the fixed end unevenness present in the beams. The timings for this case are not presented, since they simply continue the trend of the M3d matrix assembly being more costly Tahoe. The Von-Mises stresses are plotted vs. position in figure 3.7.

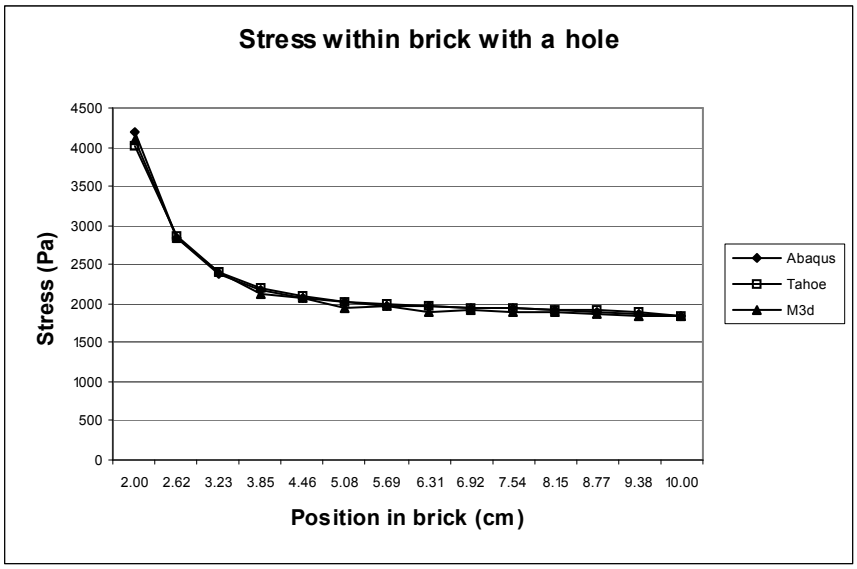

Figure 3.7. Von-Mises stresses on bottom edge of quarter brick model

4.3. Results

From the preceding examples, Tahoe appears a good replacement for the M3d solver in the program. It provides faster analyses with similar accuracy, and is capable of dealing with more complex material models and non-linear problems. The Tahoe solver module works well from within the interactive design application; the user does not realize anything has been changed, other than the results compute more quickly.

5. Conclusions

The interactive design application has been presented in its current state. The program combines computer aided design (CAD) geometry, engineering analysis results with mesh-free methods, real-time shape manipulation with fast analysis approximations, and haptic force feedback in a virtual environment for engineering design. These key potions were explained in detail.

To apply these methods to a wider variety of engineering problems and analysis types that could be encountered by a designer, the Tahoe analysis package was selected from a search of existing software options and integrated with the application. Tests were performed to compare Tahoe's mesh-free methods with those originally implemented in

the application. The results show Tahoe's speed, accuracy, and flexibility to be a favorable addition to the interactive design process.

6. Acknowledgements

The authors would like the thank the Iowa State University Virtual Reality Applications Center for the use of computational resources and hardware, and the Tahoe software developers for providing helpful advice and feedback.

7. References

1. Kim, N.H., *Design Optimization*, in *The CRC Handbook of Mechanical Engineering*. 2004.
2. Chen, J.S., et al., *A stabilized conforming nodal integration for Galerkin mesh-free methods.* International Journal for Numerical Methods in Engineering, 2001. **50**: p. 435-466.
3. Feller, J. and B. Fitzgerald. *A Framework Analysis of the Open Source Software Development Paradigm*. in *Proceedings of the twenty first international conference on Information systems*. 2000. Brisband, Queensland, Australia: ACM.
4. Sandia, N.L., *Tahoe Users guide*. 2003. Included with the Tahoe software.
5. Chipperfield, K., J.M. Vance, and A. Fischer, *Fast Meshless Reanalysis Using Combined Approximations, Pre-Conditioned Conjugate Gradient, and Taylor Series.* AIAA Journal, 2006. **44**(6): p. 1325-1331.
6. ABAQUS Inc., *What can ABAQUS do for you?*. ABAQUS website. 2005. http://www.hks.com/products/products_overview.html. July 11, 2006.
7. ANSYS Inc., *ANSYS Mechanical.* ANSYS Website. 2005. http://www.ansys.com/products/mechanical.asp. July 11, 2006.
8. Livermore Software Technology Corp, *LS_DYNA*, LSTC Website. 2005. http://www.lstc.com. July 11, 2006.
9. Wu, C.T., et al. *8th International LS-DYNA Users Conference*. 2004.

10. Shapiro, V. and I. Tsukanov, *The Architecture of SAGE - A Meshfree System Based on RFM*. 2002, Spatial Automation Library: University of Wisconsin - Madison.
11. Intact Solutions, LLC, *FieldMagic*. Intact Solutions' website. 2005. http://www.intact-solutions.com/products.php. July 11, 2006.
12. Netscape, *DMOZ Open Directory Project*. The Open Directory. 2005. http://www.dmoz.org/. July 11, 2006.
13. Forssell J., *IMPACT - a free explicit dynamic finite element program*. IMPACT dynamic finite element program suite. 2006. http://impact.sourceforge.net. July 11, 2006.
14. Dhondt, G. and K. Witting, *CALCULIX - A free software three-dimensional structural finite element program*. CALCULIX website. 2006. http://www.calculix.de. July 11, 2006.
15. Sandia National Labs, *Tahoe Development Server*. Sandia National Labs website. 2005. http://tahoe.ca.sandia.gov. July 11, 2006.
16. Demmel, J.W., J.R. Gilbert, and X.S. Li, *SuperLU Users' Guide*. 2003. Included with the SuperLU software.
17. Belytschko, T., et al., *Meshless methods: an overview and recent developments.* Computer Methods in Engineering, 1996. **139**(1-4): p. 3-47.

CHAPTER 4. STRESS SENSITIVITY CALCULATION METHODS FOR INTERACTIVE MESH-FREE REANALYSIS

To be submitted to: *Computer Modeling and Simulation Engineering*

Andrew Fischer, Judy M. Vance

1. Introduction

The evolutionary path of improving and optimizing designs is a frequent engineering task. Computational techniques such as the Finite Element Method (FEM) are common in these later stages of the design process, when the allowable changes to a product are relatively small. There is a correspondingly large body of research devoted to refining this improvement process with various optimization methods. In contrast to this, the initial creative design stage, where crucial product geometry is determined, has seen far fewer attempts at providing formal computational aid [1].

The goal of this work is to develop a methodology that aids the initial design process before a working solution exists. This is accomplished by coupling computer models with analysis models; allowing shape and design changes to be performed in real-time with fast stress analyses and re-analyses, all within a three-dimensional virtual environment [2]. A combined design and analysis environment with a minimum of restrictions on the freedom of the designer should encourage the quick investigation of many possible shape and design changes and how they affect the final product performance and operation. Obtaining a better initial design aids the evolutionary product improvement process.

Working within a virtual environment provides truly interactive design, where groups of designers and analysts work together to investigate geometry and analysis results. Mesh-free analysis methods are used in place of traditional finite elements to avoid the computationally costly re-meshing process as designers perform large shape changes. For additional information, haptic or force feedback provides another method to

aid users in determining the suitability of a design. These components work together to provide maximum freedom for the designer.

Making these interactive design methods truly practical requires that they be fast, reasonably accurate, and permit maximum freedom to explore design possibilities. These requirements are difficult to balance, as the most accurate methods are naturally the most specialized and time consuming, while the fastest approximations are often the least accurate. Working within the virtual environment becomes a dominant factor since the effectiveness of an application is shown to depend on real-time interaction [3l]. Since interactive design is the principal goal of this work, the greatest limitations are those that make these methods less interactive.

This research presents the Interactive Virtual Design Application (IVDA), a program created to implement and test these methods. The IVDA permits users to load CAD model and boundary conditions into the virtual environment, run a mesh-free analysis, and view the results. By defining a bounding volume around portions of the model, a designer may interactively change the shape and see the stress pattern update in real time using a linear Taylor series stress approximation. This approximation uses stress sensitivities computed with a finite difference technique. At any time, the model may be reanalyzed with a Preconditioned Conjugate Gradient method to update the stress pattern and sensitivities.

The time required when calculating the stress sensitivities during the reanalysis phase is one limitation to an interactive design experience. Since the stress sensitivities are currently computed via a global finite differences technique, the model must be perturbed once in each coordinate design direction and a full equilibrium analysis performed for each perturbation. This means reassembling the stiffness matrix a total of four times for each reanalysis step, once for the new stress levels and once more for each perturbation in a sensitivity direction. The finite difference technique is also very sensitive to the interval chosen for perturbation, which is currently input as a best guess by the designer. This can greatly decrease the accuracy of the calculated sensitivities. Since computing the stiffness matrix is generally very time consuming for mesh-free methods, and it must be done four times for each reanalysis, this matrix calculation is the

slowest portion of the reanalysis step, further hindering the interactive feel of these methods while in Virtual Reality (VR).

Given these limitations of the finite difference sensitivity calculation, it makes sense to pursue better methods. This work presents the results of comparing additional sensitivity calculation methods and selecting one for implementation and testing. Methods were chosen based on their applicability to the deformation techniques used in this application. The implemented method was tested based on the speed, accuracy, and robustness of the reanalysis.

The remainder of the paper is as follows: Section 2 provides an overview and sample use of the interactive design application as well as an explanation of the current sensitivity calculations. Section 3 presents the results of a literature search for sensitivity replacement methods and the selection of a method to implement. Section 4 details the implementation of the selected discreet derivatives method. And section 5 explains the tests and sample problems used to compare sensitivity calculations.

2. The Interactive Design Application

The existing application is written in C++ using the VR Juggler software library developed at the Virtual Reality Applications Center at Iowa State University. VR Juggler provides a framework for programs to run on a wide variety of virtual reality devices [4]. Currently a cluster of Linux workstations is used to run the application and drive the virtual environment. This cluster also performs the mesh-free analysis and sensitivity calculations. The application is typically used in the C6 virtual reality environment, a cubic room with six projection screens forming the six walls of the room. [5]. The user interacts with the application through a series of menus within the virtual environment, as seen in Figure 4.1.

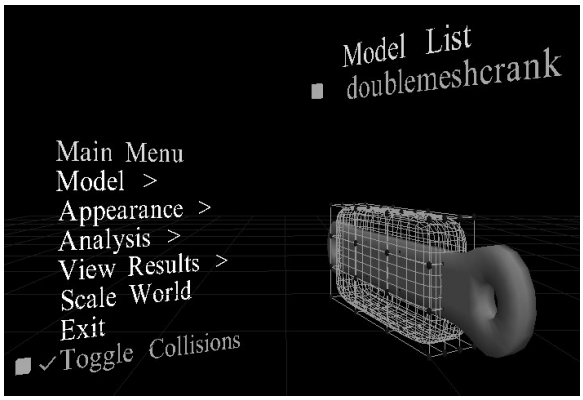

Figure 4.1. The virtual design environment.

Haptic or force feedback may also be provided to the user in the virtual environment through a SensAble Technology's PHANToM [6]. This optionally allows the user to experience forces related to the stress levels in a model while changing its shape, providing an additional channel of information about the stress state. A separate computer drives the PHANToM and networks with the cluster driving the application [7].

2.1 Using the application

The user begins by loading an existing FEA model and it's boundary conditions into the virtual environment. Models may be exported from an existing finite element package such as ABAQUS or created by hand, since a simple XML model format is used. Once loaded the model and applied boundary conditions appear in the virtual environment next to the user.

Typically a Reproducing Kernel Particle Method (RKPM) mesh-free analysis is then performed on the model to compute the displacements, stains, and stresses. In mesh-free methods the displacement approximations are given entirely in terms of the meshless nodes, so no element meshes are needed [8]. This avoids mesh distortion issues, making mesh-free particularly well suited to this application. The reproducing kernel is used to approximate unknown displacements in terms of the displacement coefficients at the mesh-free nodes. This relationship is shown in equation 1, where $u^h(\mathbf{x})$ is the displacement, $\Psi_I(\mathbf{x})$ is the reproducing kernel shape function evaluated at the point $\mathbf{x}$, with respect to the I^{th} node, and d_I are the displacement coefficients.

$$u^h(\mathbf{x}) = \sum_{I=1}^{N} \Psi_I(\mathbf{x})\ d_I \qquad \textbf{eq. 1}$$

These kernel functions are not interpolating, so the displacement at a nodal point is not equal to the value of the displacement coefficient at that node. This does make displacement boundary conditions more difficult to apply in mesh-free methods.

Two analysis options are available: 1. A custom RKPM implementation that uses Stabilized Conforming Noda Integration (SCNI) developed by Chen, et. al. and implemented at ISU [9]; 2. A mesh-free analysis using the open-source Tahoe software [10]. The use of Tahoe provides for a greater range of analysis types and options, both mesh-free and traditional FEA, and is the default analysis. The resulting stress pattern appears on the model in virtual reality along with a color bar showing the range of stress values. A variety of stress patterns may be viewed including Von Mises, maximum shear, etc. An analyzed model appears in Figure 4.2 below.

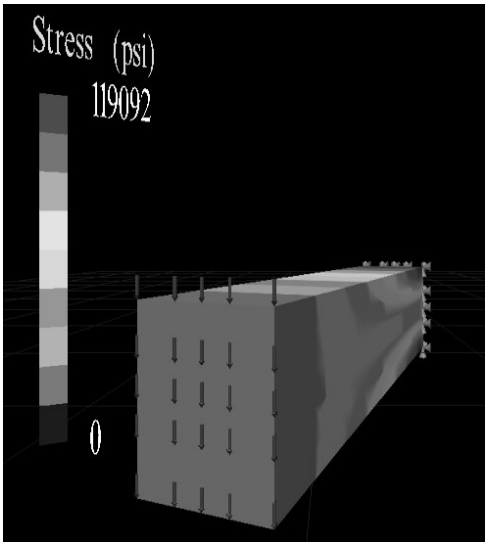

Figure 4.2. Analyzed model with boundary conditions visible

Once the stress pattern has been observed a designer will decide what portions of the model to shape change. In VR the user simply uses the hand-held wand to define one or several bounding boxes around the areas of interest. These boxes may be in turn

subdivided into smaller volumes for additional shape control. The bounding boxes are used to produce Catmull-Clark subdivision volumes, an extension of free-form deformation techniques. The model is numerically embedded in the resulting volumes which permits the user to change the model's shape by manipulating the controls points of the bounding volumes [11]. Such a bounding volume is visible around the model in the lower right portion of figure 1 above.

At this point the designer must decide what portions of the bounding volumes should be moved to change the underlying model shape. The wand is used to choose the corresponding bounding volume control points in 3D space, and the stress sensitivities are then computed based on the control points selected. This is currently performed using a finite differences technique. The control points (and therefore the model) are all deformed in the x, y, and z directions, and for each deformation, the system of equations is resolved using a PCG reanalysis method.

With the stress sensitivities computed, a linear Taylor series approximation is used to calculate the changes in stress as the model is deformed. The Taylor series approximation requires the stress (σ), a design variable (h), and the derivative of the stress with respect to that design variable, known as the stress sensitivity [12]. Using this information the new stress values (σ') may be computed as shown in equation 2

$$\sigma' = \sigma + \frac{\partial \sigma}{\partial h} \Delta h \qquad \textbf{eq. 2}$$

While this stress approximation is poor for large design changes, the simplicity of calculation makes it fast enough for interactive stress updates while the designer changes the shape. This real time update is crucial to give the designer an intuitive feel for how shape changes affect the stress contours. At any time, the user may pause moving the control points and resolve the system of equations for the deformed model. The PGC reanalysis is again used for this approximation. Stress sensitivities are again calculated using the finite differences technique, providing a new base for the Taylor series

approximation. The designer may also select different control points to change different parts of the model.

The PCG reanalysis used to resolve the system of equations is presented in detail by Chipperfield et al. [13]. In summary the method uses the solution to the initial full analysis of the mesh-free equations, which involve a linear system:

$$\mathbf{K}d = f \qquad \textbf{eq. 3}$$

In equation 3 above, **K** is the sparse stiffness matrix, d is the vector of displacement coefficients, and f is the right hand side force vector. The inverse of **K,** ($\mathbf{K}^{-1}$), is computed for the full analysis and stored. When the designer wishes to re-analyze a deformed model, a new system is formed:

$$\mathbf{K}^* d^* = f^* \qquad \textbf{eq. 4}$$

Assuming the new **K*** matrix for the modified design is similar to the initial **K** matrix, the stored $\mathbf{K}^{-1}$ matrix makes an excellent pre-conditioner to solve equation 4 with the PCG method. This technique was shown to converge quickly even for large shape changes. The PCG reanalysis is implemented with either the Tahoe software or the custom RKPM solver.

The designer is free to continue changing the part shape by moving the control points, selecting different controls points, or building new bounding volumes. Once satisfied with the resulting shape, the part may be exported back to a file for full analysis in an external finite element package. Figure 4.3 provides a sample flowchart of the program's operation.

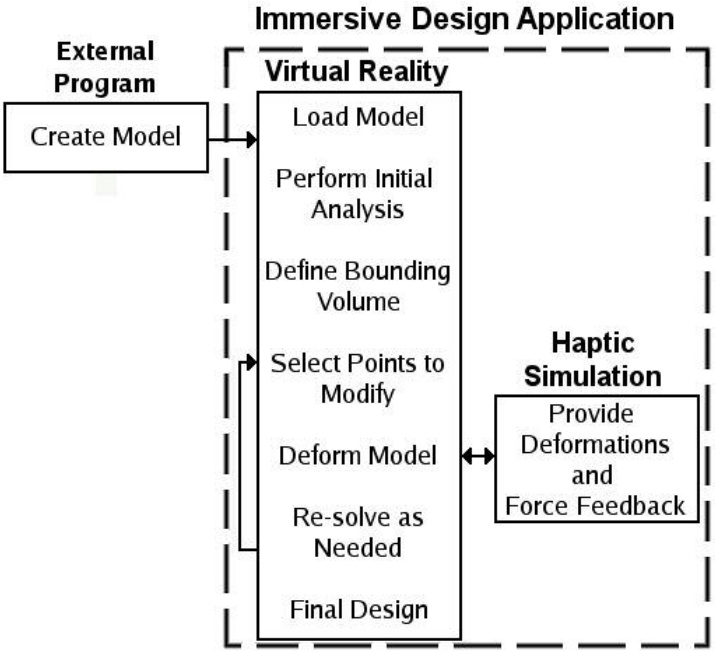

Figure 4.3: Flowchart for the interactive design application.

2.2 Calculating stress sensitivities

Stress sensitivities are a key part of the linear Taylor series approximation used to compute the stress pattern changes while the designer modifies a model in VR (see equation 2.) To maximize the effectiveness of the Taylor series approximation, the sensitivities need to be as accurate as possible. And to minimize the amount of time a designer has to wait in the virtual environment, calculation of these sensitivities should be as fast as possible.

The global finite differences technique is currently used to compute stress sensitivities every time a designer re-analyzes the model and/or chooses new control points to move. Since the designer may translate controls points anywhere in 3D space, the sensitivity values need to be computed once for each of the 3 coordinate directions (x,y,z). This requires reassembling and reanalyzing the entire system 3 times with a small perturbation in a single coordinate direction each time. If the designer has just finished moving control points to a new location, a fourth assembly and reanalysis is required to get the updated stress for the new model shape before sensitivity calculation begins. This is undesirable for two reasons:

1. It is slow. Resolving the system 4 times requires 4 different matrix assembly steps and 4 different solutions. The PGC reanalysis makes the solutions much faster, but the matrix assembly still can take quite a bit of time.
2. Accuracy is poor. The finite difference method's accuracy is highly dependent on the control point perturbation interval, and choosing the proper perturbation interval is dependent on the model being analyzed.

To overcome these limitations and determine a suitable replacement to the finite differences method, a review of other sensitivity calculation techniques was performed. The methods were then compared based on several factors, and the most promising was implemented for testing. The results of this review and the selection process for a method to implement are presented below.

3. Literature review

Kim defines design sensitivity analysis as computing "the sensitivity of performance measures with respect to design variables" [1]. In the context of the interactive design application, the performance measure relates to the stress levels in the part while the design variables relate to the changes in part shape due to free form deformation techniques used by the designer in VR.

Using the stress levels as a performance measure means we have no explicit relationship between it and the shape design variables, since in both mesh-free and FEA analyses the stress is determined from the displacement. Thus the derivative of stress with respect to the shape design variables in equation 2 is written as:

$$\frac{\partial \sigma}{\partial h} = \frac{\partial \sigma}{\partial d}\frac{\partial d}{\partial h} \qquad \textbf{eq. 5}$$

Where d is the model displacement. Since stress (σ) and displacement (d) are directly related in linear elastic analysis, it is easy to calculate the $\partial\sigma/\partial d$ term. The difficulty lies in obtaining $\partial d/\partial h$.

Following Kim's analysis, we assume our model obeys the linear equation:

$$\mathbf{K}(h)d = f(h) \qquad \textbf{eq. 6}$$

Where **K** is the stiffness matrix and f the load vector. Both are functions of and are differentiable with respect to our design parameter h, so the solution d also depends on h. By differentiating equation 6 with respect to h, we obtain:

$$K(h)\frac{\partial d}{\partial h} = \frac{\partial f}{\partial h} - \frac{\partial K}{\partial h}d \qquad \textbf{eq. 7}$$

Since we know $\mathbf{K}(h)$ and $f(h)$ we can obtain $\partial\mathbf{K}/\partial h$ and $\partial f/\partial h$, and solving equation 6 gives us d. This lets us solve for $\partial d/\partial h$ in equation 7, which may be used in equation 5 to obtain $\partial\sigma/\partial h$, the design sensitivity.

In researching design sensitivity analysis literature, it is important to note the type of design variable(s) and analysis methods being used. The interactive design application's primary purpose is to let the designer alter the shape of a model, so sensitivity analysis techniques focusing on shape design parameters is of particular interest. A mesh-free analysis is typically used to avoid mesh distortion issues when the model geometry is deformed, so literature involving other analysis types, such as the Boundary Element Method, though extensive and useful, isn't always applicable [14].

An excellent overview and comparison of the methods available for design sensitivity analysis calculation is presented by van Keulen et al [15]. There are 4 basic types of sensitivity calculation techniques available: (1) global finite differences, (2) discrete derivatives, (3) continuum derivatives, and (4) automatic differentiation. Some techniques may be further subdivided depending on whether or not analytical techniques are used when calculating various derivatives. Each technique and its applicability to

interactive design is explored below, following a brief history of the sensitivity techniques used by this application.

The earliest incarnation of the interactive design application worked on the approximation of eigenvalues and eigenvectors for large design changes. Pade approximants and curve fitting were used to achieve fast, reasonably accurate approximations, and the possibility for interactive shape changing was explored [16].

Later, sensitivity information from a pre-computed FEA analysis was taken from a commercial finite element package (MSC/NASTAN) and used to let a designer interactively alter a part shape in virtual reality [12]. When the shift was made to using mesh-free methods to avoid the element distortion errors inherent in deformed FEA models, a global finite differences approach was used due to its relatively straightforward implementation [17].

3.1 The global finite differences method

The finite differences approach is the easiest way to compute the stress sensitivity information required for interactive design. If you wish to approximate the change in some performance measure $\mathbf{P}(u)$ (the stress pattern in this application), and you know the current value of the measure at some design parameter u, you simply perturb the model shape by some small amount (Δu) and resolve the system to get $\mathbf{P}(u+\Delta u)$. The derivative may be approximated with either a simple forward or backward difference method:

$$\frac{\partial P(u)}{\partial u} \cong \frac{\left(P(u \pm \Delta u) - P(u)\right)}{\Delta u} \qquad \textbf{eq. 8}$$

Alternately, one could use perturbations in both directions resulting in a central difference technique:

$$\frac{\partial P(u)}{\partial u} \cong \frac{\left(P(u + \Delta u) - P(u - \Delta u)\right)}{2\Delta u} \qquad \textbf{eq. 9}$$

In theory higher order approximations are possible as well, though in practice they are rare for sensitivity calculation.

Advantages of the finite differences method include the ease implementation and its wide applicability. Virtually any problem that can be solved and some performance **P** calculated can be analyzed with a finite differences method. Disadvantages include the computational cost and the accuracy of the results. For each additional design variable, another solution of the equation system is required. For the interactive design application this means 3 solutions are required for every new sensitivity calculation step.

There are also two main sources of error inherent in finite differences. Both are related to the perturbation size (Δu). First is the truncation error, a result of ignoring the higher order terms in the approximation of the sensitivity ($\partial \mathbf{P}/\partial u$). The second is known as the condition error, which includes numerical and round-off errors. These sources of error tend to compete with one another, as larger step sizes increase truncation error while decreasing condition error and vice-versa. For further details see Kirsch and Bogomolni's in-depth analysis of the finite differences method and it's suitability for design sensitivity analysis [18].

3.2 Discrete derivatives

The discrete derivatives approach involves calculating portions of the differentiated discretized form of the system of equations, shown in equation 6 above. The terms that need to be found are $\partial \mathbf{K}/\partial h$ and $\partial f/\partial h$. Typically $\partial f/\partial h$ is not difficult to compute, since externally applied forces are usually either constant or not coupled to the model geometry. Calculating $\partial \mathbf{K}/\partial h$, however, depends on the type of problem being solved and how the stiffness matrix **K** was computed.

If both **K** and f may be differentiated analytically, then this is referred to as the analytical-discrete method. Such analytical differentiation, however, may be tedious to implement, especially if shape design sensitivity analysis is desired [15]. Difficulties in purely analytical differentiation are also encountered if the **K** matrix was constructed using numerical integration techniques, a common practice. To overcome these limitations, the semi analytical method may be used, which involves numerically

approximating the derivatives of **K**. A finite difference technique is typically used for such differentiation as in equation 10.

$$\frac{\partial K(h)}{\partial h} \cong \frac{\left(K(h+\Delta h) - K(h)\right)}{\Delta h} \qquad \textbf{eq. 10}$$

This is referred to as the discrete-discrete method.

Discrete discrete derivatives have the advantage of being fairly easy to implement and rather computationally inexpensive if finite differences are used to compute the derivatives of **K**. It is also a popular technique in commercial finite element software [19].

Despite it's wide use, there are disadvantages to the discrete derivatives approach. The analytical version is simply not practical to implement for shape design sensitivity analysis of a wide range of models. The discrete method, though popular, has been shown to exhibit accuracy problems in shape design analysis for a variety of structures [20].

To remedy this problem, Lund et al proposed a method for the exact numerical differentiation of the element matrices. By improving the approximate numerical derivatives of the matrices may be upgraded to exact differentiation by the application of correction factors. The results are supposed to eliminate the accuracy issues associated with the semi-analytical method [21]. Their work also contains a detailed implementation of the method with 3D solid elements.

3.3 Continuum derivatives

The continuum method takes a different approach at a higher mathematical level than the other techniques listed here. The differentiation is applied to the variational equations, before the discretization of the model into a system of equations. For a 3D elastic solid, as is commonly used in the interactive design application, the variational equation may be expressed as:

$$\iiint_{\Omega} \sigma(z) : \varepsilon(z) d\Omega = \iiint_{\Omega} \bar{z}^{T} f d\Omega + \iint_{\Gamma} \bar{z}^{T} \sigma(z) n d\Gamma \qquad \textbf{eq. 11}$$

By defining bilinear and load linear forms for the left and right hand sides of equation 11 respectively, it may be simplified to:

$$a(z,\bar{z}) = l(\bar{z}) \qquad \textbf{eq. 12}$$

The terms in equation 12 are then differentiated with respect to the design parameters of interest, and the resulting equations are solved to obtain the structural sensitivity. This process is mathematically quite involved, and is well presented by Choi and Kim in their comprehensive book on structural design sensitivity analysis [22]. Their work is used as the main reference for continuum derivative design sensitivity analysis in the remainder of this paper.

The continuum method may be further subdivided into two types, depending on whether exact solutions to the continuum equations exist or not. If they do exist and are used the calculation is a continuum-continuum method, and it provides an exact analytical expression for the design sensitivity analysis. However, such exact solutions exist for a very small subset of problems. Far more useful for practical engineering problems is the continuum-discrete method, where FEA methods are used to compute the design sensitivity via the differentiated continuum equations.

Advantages of the continuum methods are the evaluation of accurate design sensitivity information without recourse to any sort of finite difference technique and it's associated uncertainties. Continuum methods are also developed independent of any particular analysis method, so they apply equally well to standard FEA and mesh-free techniques.

One potential drawback to the continuum methods as used in this paper is the need to calculate the design velocity field associated with the shape changes in a model due to the designer's actions. The accuracy of the sensitivity results depends on the accuracy of the design velocity field calculation, and the design velocity field must meet certain requirements. Choi and Kim present several alternatives to compute the design velocity field, including a finite difference method, an isoparametric mapping technique,

a boundary displacement method, and a hybrid method. To use continuum sensitivity methods with the interactive design application one would have to ensure the implemented design velocity field calculation method works with the free-form deformation techniques used.

3.4 Automatic differentiation

Automatic or algorithmic differentiation refers to computing the derivatives of functions in the computer analysis code itself. Since computer analysis codes are really just a collection of elementary functions, it is possible to define partial derivatives of these elementary functions and related subroutines using the chain rule of differentiation. Software packages exist to perform this automatic differentiation, such as ADOL-C [23], ADIC [24] and the open source CppAD [25] for C++ computer code.

Given a computer function $F:R^n->R^m$, the automatic differentiation software would produce a fuction F' that not only evaluates F for any input $x \in R^n$, but also it's Jacobian $J \in R^{mxn}$ at the same value of x. The software continually applies the chain rule of differential calculus to the simplest mathematical operators in the computer code, eg addition, multiplication, trig functions, etc., that have known derivatives. These individual derivatives are combined to yield the derivative of the entire function.

Automatic differentiation also may be applied via two distinct approaches. The forward mode works with the function output variables where derivatives are to be computed, called dependent variables. Input variables to the function being differentiated are independent variables. Derivatives are propagated along with the control flow of the function F. The reverse mode produces derivatives by running the function and it's subroutines in reverse [26]. In general, the forward mode requires more CPU time but lower memory storage, while the reverse mode is favorable from a computational standpoint but can require significantly more computer memory since it actually runs two sweeps through the function [27].

Despite its name, automatic differentiation is not always purely automatic. It is sometimes necessary to tweak the results of applying automatic differentiation software to an analysis code to increase efficiency. Still, automatic differentiation is finding

increased use in optimization related fields. To date, applications consisting of around 400,000 lines of code have been differentiated [28]. A collaborative effort between Argonne National Laboratory and RWTH Aachen University has created www.autodiff.org, a comprehensive web site with information about the process of auto-differentiation.

3.5 Considerations for implementation

Selecting the sensitivity calculation method to implement and compare was done by choosing 5 criterion and evaluating each method's performance based on that criterion. These criterion are somewhat different for the interactive design application than a more traditional analysis program due to the focus here on early stage design, speed of calculation, and more use of qualitative results to aid designers who have been given great freedom to interactively change geometry. The criterion selected were:

1. Expected speed improvements. The primary reason for seeking alternative sensitivity methods to global finite differences is the long time required for calculations; three full matrix assemble and solve steps. An alternative method should be faster by avoiding additional assemble-solve steps.
2. Perturbation interval dependence. The finite difference method's strong dependence on perturbation interval size reduces the sensitivity accuracy. A replacement method should reduce or eliminate this dependency.
3. Ability to use the selected method with shape design sensitivity analysis through subdivision surface free-form deformation. The interactive virtual design application's use of free-form deformation for shape design is an important tool, since it provides designers nearly unlimited freedom to change shape and size in virtual reality. It has to be applicable to the selected sensitivity method, as when computing the design velocity field.
4. Applicability of the selected method to mesh-free analysis methods. Mesh-free methods offer improved solution accuracy over traditional finite element methods when large shape changes are taking place. Though the shape design sensitivity

technique chosen should be general enough to apply to both analysis methods, a method with examples of a mesh-free implementation would be very helpful.

5. Ease of implementation within the framework of the existing application. The selected method should be easy to plug into the existing application structure without substantial rewriting. It should also have the potential to be easily removed or exchanged for another method.

Each of the sensitivity analysis methods listed in sections 3.1-3.4 were compared based on the above criterion. A summary of the results appears below on a per method basis.

Global Finite Differences

The current finite difference technique has several strong advantages, which is why it was implemented in the first place. It is equally easy to apply to shape design sensitivity analysis as to any of the other design parameters, it applies equally well to mesh-free and standard finite element methods, and it functions nearly independently of the method used to deform the model, since you only need nodal locations before and after deformation without regard to how they were moved. As mentioned, however, for speed and accuracy considerations the finite differences technique is considered unacceptable in the context of the interactive design application.

Discrete derivatives - semi analytical method

Since the full analytical method for discrete derivatives requires analytical differentiation of **K** and **f**, it is considered impractical to implement in light of the requirements for shape design sensitivity analysis and the numerical methods used to assemble the element matrices.

The semi analytical method has the advantage of being widely used so reference implementations are easy to locate. It's more efficient than finite differences so it is expected to be faster. It requires design velocity field information to be computed with

the subdivision surface techniques, which is similar to the methods used in the semi analytical method literature. The theory is general enough to apply to mesh-free methods.

The accuracy errors exhibited in shape design analysis are troubling, but the exact numerical differentiation method proposed by Lund et al is very promising. It removes the perturbation size dependence and claims excellent accuracy. Implementation should be straightforward in light of the presented examples with 3D solid elements.

Continuum derivatives

The continuum derivative method presented by Choi and Kim is a good candidate for several reasons. Since differentiation takes place before discretization, there is no need to compute stiffness matrix derivatives, so the results are expected to be quite accurate. The method also applies well to all analysis techniques, since it is developed independently of the analysis method.

Mathematically the continuum methods are more involved than the other techniques listed here. The speed of calculation may be slightly slower since some additional steps are required vs. the discrete method. There is also less literature available on their implementation, since they appear more recently developed and less widely understood. The continuum method also places strict requirements on the computation of the design velocity field, which may not be guaranteed by the subdivision volume deformation techniques used here.

Automatic differentiation

The theoretical advantage of automatic differentiation is it's broad applicability to an existing analysis code and it's accuracy. It should also be unrelated to the analysis method and the geometry deformation technique used. Studies show automatic differentiation to be very computationally efficient, so it is expected to be very fast.

Practically, use of automatic differentiation may not always be automatic or easy, especially for large analysis codes. The methods also seem fairly experimental yet, with few details of large scale general implementations available. It would also require a substantial number of source code changes to the existing program and to Tahoe, instead

of making an additional code module to handle the sensitivity analysis. This would greatly increase the implementation difficulty.

To help visualize the differences between methods for shape design sensitivity analysis, a simple decision matrix was created. The matrix appears in table 1 below. The different criterion appear across the top while the choice of analysis technique appears down the left column. Each technique was rated on a scale from 1-3 for each criterion with 4.1 being the worst and 3 the best. The finite difference technique is included here for reference.

	Speed	Accuracy	*Shape Design Subd. Volume*	*Meshfree Applicability*	*Ease of Implementation*	***Total***
Finite Differences	1	1	3	3	3	**11**
Discrete Derivatives	2	3	3	3	3	**14**
Continuum Derivatives	2	3	2	3	2	**12**
Automatic Differentiation	3	3	3	3	1	**13**

Table 4.1. Decision matrix for selecting a sensitivity calculation method.

From the decision matrix, it appears that the discrete derivatives come out ahead, with the automatic differentiation method just behind. In light of this, and the excellent documentation presented by Lund et al, the discrete derivatives method with exact numeric differentiation was selected for implementation and testing vs. the existing finite differences sensitivity calculation method. Section 4 details this implementation.

4. Implementation

The interactive design application itself is a collection of C++ modules that are linked together to form the overall program. The modules communicate with each other via set methods, which makes it easy to add/remove/change modules without affecting the rest of the application. The main module uses VR Juggler to drive the virtual environment and receive user input. Models are loaded from disk via the I/O manager and controlled by the model manager. Either the Tahoe analysis software or a custom mesh-free solver (M3D) may be used to perform the actual analysis, while a separate module handles the PCG reanalysis. The Taylor Series approximation for stresses computes the updated stress contours based on stress sensitivity information as a model is altered via free form deformation.

A diagram of the program layout appears in figure 4. Solid ovals denote modules written specifically for the interactive design application while dotted denotes 3rd party software used by the application.

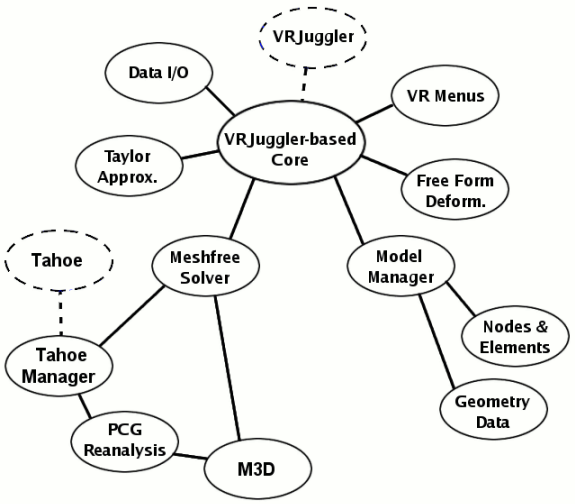

Figure 4.4. Diagram of the Interactive Design Application

In this design, the stress sensitivity calculations are performed by the main module since the actual finite difference calculation is quite simple. Information on stress states from the solver is used with nodal position data from the model to generate the sensitivity data for the Taylor Series.

To work with more advanced stress sensitivity calculation methods, a separate shape design sensitivity analysis (SDSA) module was created and abstracted out of the main module. It is general enough to work with data from either the custom M3D solver or the more powerful Tahoe analysis software. Since implementing the discrete derivatives with exact numeric differentiation required more information from the analysis software, the SDSA module was made a sub module of the solver, with access to special information from either M3D or Tahoe.

4.1 Discrete derivatives with exact numeric differentiation

The basic tenet of discrete derivatives is the implicit differentiation of the overall equilibrium equation, resulting in equation 7 above. It is repeated here as equation 13 for convenience.

$$K(h)\frac{\partial d}{\partial h} = \frac{\partial f}{\partial h} - \frac{\partial K}{\partial h}d \qquad \textbf{eq. 13}$$

As mentioned, the calculation of $\partial\mathbf{K}/\partial h$ is the difficult part in this process. $\partial f/\partial h$ is simple, and vanishes for design independent loads. $\partial\mathbf{K}/\partial h$ is typically calculated and assembled on an element-by-element stiffness matrix (**k**) basis, as in equation 14, in much the same way as the global stiffness matrix (**K**).

$$\frac{\partial K}{\partial h} = \sum_i \frac{\partial k_i}{\partial h} \qquad \textbf{eq. 14}$$

Only the elements perturbed by the model deformation will contribute to the overall calculation of $\partial\mathbf{K}/\partial h$. So while equation 14 sums over all elements, those elements that remain un-deformed may be skipped.

Calculating the derivatives of each element stiffness matrix (**k**) requires working with the deformed coordinates of the element. The deformed element coordinates may be obtained using the shape design change and the chain rule in equation 15.

$$\frac{\partial k}{\partial h} = \sum_j \frac{\partial k}{\partial n_j} \frac{\partial n_j}{\partial h} \qquad \textbf{eq. 15}$$

The $\partial n/\partial h$ term is the design velocity field, representing the change in element coordinates with respect to the change in subdivision volume control point positions. This is obtained via finite differences since there is no explicit relationship between the two for the subdivision volume deformation methods used in the interactive design application.

To obtain the $\partial \mathbf{k}/\partial n_j$ term in equation 15, the exact numeric differentiation technique of Lund et al is applied, starting with the expression for the element stiffness matrix **k**.

$$k = \int_\Omega \left(B^T E B |J| \right) d\Omega \qquad \textbf{eq. 16}$$

Here **B** is the strain displacement matrix, **E** the element material matrix, and **J** the Jacobian. Differentiating **k** with respect to the element coordinates (n) gives the following:

$$\frac{\partial k}{\partial n} = \int_\Omega \left(\frac{\partial B^T}{\partial n} E B + B^T E \frac{\partial B}{\partial n} |J| \right) d\Omega + \int_\Omega B^T E B \frac{\partial |J|}{\partial n} d\Omega \qquad \textbf{eq. 17}$$

Two derivatives are now needed, $\partial \mathbf{B}/\partial n$ and $\partial |\mathbf{J}|/\partial n$. Since the coordinates x,y and z only appear linearly in the determinant of the Jacobian, $|\mathbf{J}|$, the derivative of $|\mathbf{J}|$ will be independent or at most linear with respect to nodal coordinates n. Hence $\partial |\mathbf{J}|/\partial n$ may be computed exactly via a forward finite difference scheme.

$$\frac{\partial |J(n)|}{\partial n} \cong \frac{\left(|J(n + \Delta n)| - |J(n)| \right)}{\Delta n} \qquad \textbf{eq. 18}$$

The $\partial\mathbf{B}/\partial n$ term is more complicated, since it requires the shape function derivatives $\partial\mathbf{N}/\partial n$. The strain displacement matrix **B** is calculated from each of the nodal coordinates as in equation 19.

$$\frac{\partial B}{\partial n} = \left[\frac{\partial b_1}{\partial n} \frac{\partial b_2}{\partial n} \cdots \frac{\partial b_j}{\partial n} \right] \qquad \textbf{eq. 19}$$

Here j is the number of nodes. Each nodal $\partial b/\partial n$ term is a matrix of shape function derivatives:

$$\frac{\partial b}{\partial n} = \begin{bmatrix} \frac{\partial N_{i,x}}{\partial n} & 0 & 0 \\ 0 & \frac{\partial N_{i,y}}{\partial n} & 0 \\ 0 & 0 & \frac{\partial N_{i,z}}{\partial n} \\ \frac{\partial N_{i,y}}{\partial n} & \frac{\partial N_{i,x}}{\partial n} & 0 \\ 0 & \frac{\partial N_{i,z}}{\partial n} & \frac{\partial N_{i,y}}{\partial n} \\ \frac{\partial N_{i,z}}{\partial n} & 0 & \frac{\partial N_{i,x}}{\partial n} \end{bmatrix} \qquad \textbf{eq. 20}$$

Using the Jacobian matrix and finite differences, the shape function derivatives $\partial N_{i,(x,y,z)}/\partial n$ may be found.

$$\frac{\partial}{\partial n} \begin{Bmatrix} N_{i,x} \\ N_{i,y} \\ N_{i,z} \end{Bmatrix} = -J^{-1} \frac{J(n+\Delta n) - J(n)}{\Delta n} \begin{Bmatrix} N_{i,x} \\ N_{i,y} \\ N_{i,z} \end{Bmatrix} \qquad \textbf{eq. 21}$$

Now, all terms needed to form the exact derivative of the element stiffness matrix are computed. They can be used to form $\partial \mathbf{K}/\partial h$ and solve for the $\partial d/\partial h$ term in equation 13. Once $\partial d/\partial h$ is found, the stress sensitivities, $\partial \sigma/\partial n$**,** may be calculated using equation 22.

$$\frac{\partial \sigma}{\partial n} = E\left(\frac{\partial B}{\partial n} d + B\frac{\partial d}{\partial n}\right) \qquad \textbf{eq. 22}$$

4.2 Design sensitivity analysis with Tahoe

The Tahoe software itself currently does not provide any support for doing design sensitivity analysis calculations. Since Tahoe was shown to be a faster, more accurate replacement for the original M3D solver in this application, performing Design Sensitivity Analysis (DSA) with Tahoe is an important goal. While applying the global finite differences technique is a trivial process, the discrete derivatives method with exact numeric differentiation was more challenging.

Tahoe is a large project with support for many different elements, material models, and analysis types. It supports mesh-free methods, crack analysis, cohesive models and a number of other more specialize features [29]. Fortunately Tahoe is well structured and makes generous use of the C++ language's advanced features, so the changes necessary to add a DSA module can be done cleanly and with a minimum amount of code.

To obtain the element information necessary to perform discrete derivative DSA, accessor methods were added to certain Tahoe classes. Code was added to base element types to return the Jacobian and strain displacement matrices, as well as compute their numeric derivatives with respect to coordinate changes. Any changes made to the Tahoe source code were stored as a series of patch files, so they may be easily applied to a new version of the software.

A custom version of the Tahoe FEManager, used to tie it the analysis in with the virtual design application, was given extra functionality as well. It was used to assemble the element stiffness matrix derivatives and solve for the displacement derivatives using

the already factored and stored **K** matrix. The shape design sensitivity analysis program module uses this information to bring stress sensitivity information to the deforming model thereby generating improved Taylor series stress approximations.

5. Example problem

To compare the speed of the discrete derivatives method with the global finite differences technique, an example problem is presented. A simple 3D beam, dimensions 2cm x 2cm x 10cm, is fixed on one end and placed under a unit load at the other. Around the center, a 12 control point subdivision volume is positioned. The top two control points are selected and stress sensitivities are computed for the manipulation of those two control points in the x,y and z coordinate directions. Figure 4.5 details the example setup. This particular problem was small, only 625 elements.

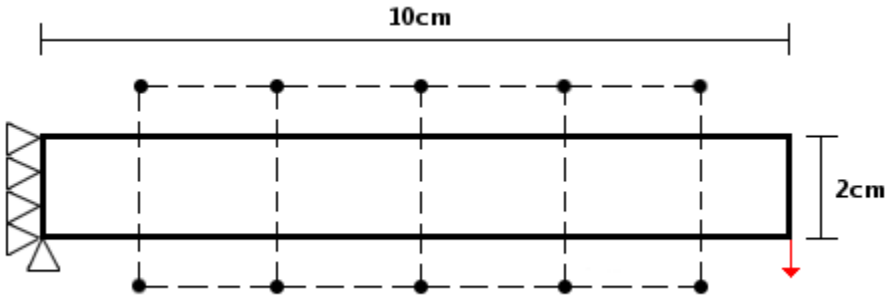

Figure 4.5. Beam example diagram

Since the interactive design application allows the sensitivity calculation method to be selected on the fly, the model was loaded once and analyzed. This generates the factored stiffness (**K**) matrix, which is stored for repeated sensitivity calculations. The finite differences method was used first, followed by the discrete derivatives technique. Each method was run 5 times, and the timing results appear below in Table 4.2.

Run	Finite Differences	Discrete Derivatives
1	2.43723	0.486943
2	2.43421	0.447847
3	2.4377	0.40741
4	2.43462	0.405784
5	2.42925	0.448757
6	2.43647	0.447968
7	2.44751	0.450565
8	2.43513	0.490276
9	2.42954	0.486614
10	2.44616	0.40584
Average	2.436782	0.4478

Table 4.2. Comparison of times (in seconds) required to compute sensitivity values for a model with 625 elements

As expected, the discrete derivatives method is noticeably faster than the finite differences technique for this problem. Due to the small number of elements, the repeated matrix assemble and solve steps used in the finite difference technique end up being quite a bit more costly than the discrete derivatives.

To gauge the effect of larger problems on timings, the above sample problem was repeated for a model discretized with 1617 elements. All other conditions were the same. The times for this case appear in Table 4.3.

Run	Finite Differences	Discrete Derivatives
1	8.80334	3.16602
2	8.82466	3.20179
3	8.79284	3.11785
4	8.81535	3.07922
5	8.80418	3.16653
6	8.81807	3.13082
7	8.80089	3.08936
8	8.82984	3.04443
9	8.79877	3.04512
10	8.83926	3.04168
Average	8.81272	3.108282

Table 4.3. Comparison of times (in seconds) required to compute sensitivity values for a model with 1617 elements

Lastly, a graph was made to illustrate the difference in stress sensitivity calculations. For the 1617 element beam, the Y-direction (upwards) stress sensitivity value was extracted for each element (or mesh-free node) in a strip along the top center of the beam. The plot appears in figure 4.6.

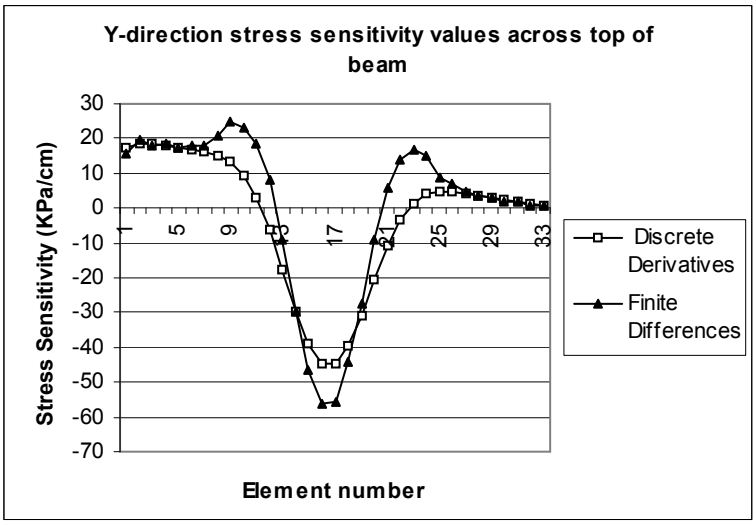

Figure 4.6. Plot of Y-direction stress sensitivity values across top center of beam

The above plot shows some qualitative differences between the two methods. Both give the expected behavior, since stress is expected to decrease in the mid-section as you raise the height, thus thickening the beam through the cross section. The discrete derivative method, however, provides a smoother solution with less extreme values.

6. Conclusion

To achieve faster, more accurate interactive design, the sensitivity calculations for Taylor Series stress approximations were improved. A review of the available shape design sensitivity methods was made, and a discrete derivatives technique with exact numeric differentiation was selected. This technique was implemented with the Open Source Tahoe analysis software and tested on a sample problem.

7. Acknowledgements

The authors would like to thank the Tahoe software developers for insight and help with questions, and the Iowa State University Virtual Reality Applications Center for the use of computational resources and hardware.

8. References

1. Kim, N.H., *Design Optimization*, in *The CRC Handbook of Mechanical Engineering*. 2004.
2. Chipperfield, K.A., *Interactive stress re-analysis in virtual reality*, in *Mechanical Engineering*. 2002, Iowa State University: Ames.
3. Jayaram, S., et al., *Assessment of VR Technology and its Applications to Engineering Problems.* Journal of Computing and Information Science in Engineering, 2001. 1: p. 72-83.
4. Bierbaum, A., et al. *VR Juggler: A Virtual Platform for Virtual Reality Application Development*. in *Virtual Reality, 2001*. 2001. Yokohama, Japan: IEEE.
5. Virtual Reality Applications Center, *About the C6*. VRAC Intranet. 2001. https://intranet.vrac.iastate.edu/vr/c6/info/page.html. July 16, 2006.
6. Fischer, A. and J.M. Vance. *PHANToM Haptic Device Implemented in a Projection Screen Virtual Environment*. in *7th International Immersive Projection Technologies Workshop*. 2003. Zurich, Switzerland.
7. Kim, C. and J.M. Vance. *Development of a networked haptic environment in VR to facilitate collaborative design using Voxmap Pointshell (VPS) software*. in *ASME Design Engineering Technical Conferences and Computer and Information in Engineering Conference*. 2004. Salt Lake City, Utah.
8. Hardee, E., et al., *A Structural Nonlinear Analysis Workspace (SNAW) based on meshless methods.* Advances in Engineering Software, 1999. 30: p. 153-175.
9. Chen, J.S., et al., *A stabilized conforming notal integration for Galerkin mesh-free methods.* International Journal for Numerical Methods in Engineering, 2001. 50: p. 435-466.
10. Sandia National Labs, *Tahoe Development Server*. Sandia National Labs website. 2005. http://tahoe.ca.sandia.gov. July 11, 2006.
11. MacKracken and Joy. *Free-Form Deformations With Lattices of Arbitrary Topology*. in *SIGGRAPH 96*. 1996.

12. Yeh, T.-P. and J. Vance, *Applying Virtual Reality Techniques to Sensitivity-Based Structural Shape Design.* ASME Journal of Mechanical Design, 1998. 120(4): p. 612-619.
13. Chipperfield, K., J.M. Vance, and A. Fischer, *Fast Meshless Reanalysis Using Combined Approximations, Pre-Conditioned Conjugate Gradient, and Taylor Series*. AIAA Journal, 2006.
14. Calvo, E. and L. Gracia, *Shape design sensitivity analysis in elasticity using the boundary element method.* Engineering Analysis with Boundary Elements, 2001. 25: p. 887-896.
15. van Keulen, F., R.T. Haftka, and N.H. Kim, *Review of Options for Structural Design Sensitivity Analysis. Part1: Linear Systems.* Computer Methods in Applied Mechanics and Engineering, 2004. accepted.
16. Vance, J.M. *Approximating Eigenvalues and Eigenvectors Using Pade Approximants*. in *ASME Design Automation Conference*. 1993. Albuquerque, NM: ASME.
17. Chipperfield, K., T.-P. Yeh, and J.M. Vance. *Interactive product development in a virtual environment utilizing haptics*. in *2002 NSF Design, Service and Manufacturing Grantess and Research Conference Proceedings*. 2002. San Juan, Puerto Rico.
18. Kirsch, U. and M. Bogomolni, *Efficient Finite Difference Design Sensitivities.* AIAA Journal, 2005. 43(2): p. 399-405.
19. ABAQUS Inc., *ABAQUS Users Guide*. 2005. Included with the ABAQUS software.
20. Barthelemy, B. and R.T. Haftka. *Accuracy analysis of the semi-analytical method for shape sensitivity calculation*. in *The 29th Structures, Structural Dynamics and Materials Conference*. 1988. Williamsburg, VA: AIAA.
21. Lund, E., et al., *Shape Design Sensitivity Analysis in Solid Modeling Base CAD Systems*. 1996, University of Pavia: Pavia, Italy.
22. Choi, K.K. and N.H. Kim, *Structural Sensitivity Analysis and Optimization*. 2005, New York, NY: Springer Science+Business Media, Inc.

23. Walther, A. and A. Griewank, *ADOL-C A Package for Automatic Differentiation of Algorithms Written in C/C++*. 2006. Technische Universität Dresden Math department website. http://www.math.tu-dresden.de/~adol-c/. July 21, 2006.

24. University of Chicago, *ADIC Resource Center*. Argonne National Laboratory website. 2003. http://www-fp.mcs.anl.gov/adic/. July 21, 2006.

25. Bell, B., *CppAD: A Package for C++ Algorithmic Differentiation*. COIN-OR Foundation website. 2006. http://www.coin-or.org/CppAD/. July 21, 2006.

26. Bischof, C.H., et al., *Efficient and accurate derivatives for a software process chain in airfoil shape optimization.* Future Generation Computer Systems, 2005. 21: p. 1333-1344.

27. Tolsma, J.E. and P.I. Barton, *On Computational Differentiation.* Computers in Chemical Engineering, 1998. 22(4/5): p. 475-490.

28. Aachen and Argonne, *Autodiff.org*, The Automatic Differentiation website. 2006. http://www.autodiff.org. July 21, 2006.

29. Sandia National Labs., *Tahoe Users guide*. 2003. Included with the Tahoe software.

CHAPTER 5. HAPTIC FEEDBACK TO GUIDE INTERACTIVE DESIGN

For submission to: *Presence: Teleoperators and Virtual Environments*

Andrew Fischer, Dao M. Vo, Judy M. Vance

1. Introduction

Virtual Reality (VR) allows engineers to naturally interact with three-dimensional digital models in a three-dimensional space. This provides a unique interface between users and computer models not found in traditional desktop environments. Certain areas of engineering design use virtual reality in many ways, from prototype evaluation and virtual assembly to visualizing volumetric data sets [1] [2].

This work is based on a methodology for interactive design that uses virtual reality to aid the engineering design process. Computer analysis models with fast reanalysis approximations are coupled to geometric models in a virtual environment, permitting shape design changes and updated analysis results in real-time. This combined design and analysis environment places a minimum number of restrictions on the designer, encouraging the rapid investigation of many possible shape and design changes and how they affect the final product. The application developed to implement these ideas is referred to as the Immersive Virtual Design Application (IVDA).

To make working in such a virtual environment an effective experience, the user must feel immersed in the application. Immersion refers to a sense of "being there" that a user feels in the virtual world; the greater the level of immersion, the more real the virtual world appears and the more useful it becomes [3]. The level of immersion experienced in VR ranges from simple stereo vision on a desktop computer monitor to a multi-screen projection environment complete with active stereo vision, user position tracking and surround sound. The perceived level immersion in a VR environment is directly related to the number of senses stimulated, such as sight and hearing [4].

Unfortunately, many virtual reality systems and applications lack a key area of sensory stimulation: some form of physical force or haptic feedback. The word haptics refers to the feeling of force, weight, roughness or other physical resistance felt by a user in a virtual environment. Adding haptic feedback to a virtual environment is expected to improve the level of immersion and thus the effectiveness of the application. Investigation of task times for virtual assembly indicates that adding force feedback increases the efficiency of the application [5]. Similarly, virtual prototyping, where virtual reality is used to evaluate part designs for criteria such as ease of use by human operators, also indicates the addition of haptic feedback significantly decreases task completion times [6].

The goal of this paper is to present the integration of haptic feedback to the existing interactive design application and to explore different ways to convey information about the analysis back to the designer. The remainder of this paper is a follows: Section 2 reviews literature about haptics and haptic devices, and outlines the immersive virtual design application. In section 3 the implementation of haptic feedback with the design application is detailed. Section 4 discusses different ways to present information to the user via haptic feedback. And section 5 covers the results of a pilot study run to gage the effectiveness of different types of haptic feedback for users of the immersive virtual design application.

2. Background

The word "haptic" comes from the Greek haptesthai, meaning to touch or grab. Research shows that the sense of touch actually has two components, tactile and kinesthetic. Tactile refers to the actual touching of a surface and the sensing of roughness, temperature, etc. Kinesthetic (dynamic) touch provides information about the physical properties of a whole object such as weight, size, and inertia. While the tactile sense depends on nerve endings in the body, the kinesthetic relies on the position of, and forces applied to, a user's hand and limbs [7].

2.1 Haptic devices

Since a designer is expected to use whole arm motions and movement to work with the immersive design application, the decision was made early on to only explore kinesthetic haptic devices. Details on tactile devices are beyond the scope of this paper. A review of several kinesthetic haptic devices considered for use with the IVDA follows.

2.1.1 The PHANTOM

Developed by SensAble Technologies, the PHANTOM family is a popular group of haptic devices for commercial and research applications. Originally, the PHANTOM was developed as a high fidelity, low inertia and relatively low-cost device to provide force interaction with virtual objects [8]. Today a number of different PHANTOM models are available, from the small desktop OMNI to the larger, more powerful 3.0 [9]. A picture of the PHANTOM 3.0 in use appears in figure 5.1 below.

Figure 5.1. The PHANTOM 3.0
Courtesy of SensAble Technologies

To program for and control the PHANTOM two different software toolkits are available. The GHOST software is a collection of C++ classes that provide a scene graph interface to working with the PHANTOM. It does of most of the low-level work required to operate the PHANTOM, including the creation and maintenance of a separate haptic process to control the motors and keeps the force update rates within the proper range [10]. The newer OpenHaptics toolkit provides a high level interface to haptics

programming with a structure similar to the OpenGL API. Written in C++, OpenHaptics also provides routines for lower level device control [11].

2.1.2 The CyberForce

The Immersion Corporation provides a number of haptic devices designed around an exoskeleton for the users hand known as the CyberGrasp. This allows forces to be applied to each finger individually to simulate grasping and holding [12]. The CyberGrasp is operated from a backpack worn by the user, making the device portable.

However, the CyberGrasp alone is limited in that it does not allow the user to feel weight, mass, or forces against the entire arm. To remedy this, the CyberForce is offered. It combines the CyberGrasp exoskeleton with a large PHANTOM-like device that provides grounded force feedback for the whole arm. This lets the CyberForce simulate grasping and moving objects in a very realistic fashion [13]. A picture of the CyberGrasp glove appears in Figure 5.2.

Figure 5.2. The CyberGrasp glove
Courtesy of the Immersion Corporation

2.1.3 The DELTA device

DELTA haptic devices are provided by the Swiss company Force Dimension. They are based on a parallel mechanism design that provides higher continuous forces

and stiffness levels than other devices, but at the cost of a smaller physical workspace [14]. A number of different DELTA devices are offered, including a 6 degree of freedom model [15]. A software toolkit is also provided to customers for controlling and programming the devices. The delta haptic device appears in figure 5.3.

Figure 5.3. The DELTA haptic device
Courtesy of ForceDimension

2.2 Uses for haptic technology

There are a variety of fields that use haptic devices in situations such as design, simulation and operation. These range from more traditional areas such as surgery and assembly tasks to the more exotic exploration of multidimensional data sets.

In the early stages of product design, models, such as automotive body shapes, are often created using clay and then developed into a CAD model. Obtaining this CAD model is a time consuming and inexact process. As a work around, a "digital clay" program that uses the PHANTOM to let designers sculpt clay models on the computer with a variety of tools has been developed [7]. SensAble's FreeForm modeling system, a digital clay sculpting software package for industrial designers, combines clay's or foam's ease of use with haptic feedback and the advantages of a digital model allowing for shorter product development times [16].

Medical surgery sees several benefits from haptic technology, especially surgical training. Simulators using one or more haptic devices to represent surgical instruments not only provide feedback for simulating bodily tissues but can also record the surgeons' actions to monitor their skill and progress [17]. Laparoscopic surgery, where the surgeon's work is done through small incisions in the skin and guided by images from a fiber optic camera, lends itself well to haptic aid. Since the actual surgery is performed by watching a computer display, adding haptic feedback to simulator tools with detailed physical modeling gives a highly realistic simulation [18].

At Boeing, researchers have integrated a six-degree of freedom PHANTOM with their Voxmap PointShell (VPS) collision detection software to create a virtual assembly tool. VPS allows a polygonal model to be discretized into a collection of voxels, which can be used as the basis for a very fast collision detection algorithm [19]. This allows for the dynamic manipulation of a detailed rigid object within an environment whose complexity is only limited by computer memory, all while maintaining haptic update rates [20].

Rakesh Gupta et al developed a design for assembly analysis tool called the Virtual Environment for Design for Assembly or VEDA to investigate the effectiveness of virtual assembly simulations. Haptic feedback using two PHANTOMs, physically based modeling, accurate collision detection, and sound cues all provide a realistic virtual assembly experience. Tests compared the task completion times for an actual assembly process and an identical virtual assembly process. Results indicated VEDA assembly times correlate with actual assembly times as task difficulty increased, though all virtual task completion times were roughly twice as long [1].

To help make virtual assembly a more practical design tool, Howard and Vance present an assembly application that uses desktop VR with physically based modeling. The Open Physics Abstraction Layer is used to provide a stable physics backend while the PHANTOM Omni gives effective, low cost haptic feedback for mechanical assembly [21].

Haptics technology is also common in dataset visualization. Avila and Sobierajski explore the use of a PHANTOM with haptic feedback as an input and output device for

exploring complex three-dimensional data. They report haptic integration gives the user a more intuitive method to understand and alter such data [2].

Researchers at Iowa State University have been working with a haptic enabled device called the Reachin Display for interaction with and display of 3D data. Their work takes advantage of the additional feedback available through haptics and sound to aid environmental planning within a Geographic Information System. Their goal is to use this additional information to guide users to an optimal solution in the planning process, similar to the goals set forth for the IVDA in this paper [22]. The Reachin Display by Reachin Technologies uses a PHANTOM haptic device and a semi-transparent mirror to produce a co-located display where a user sees and feels an object in the same location [23]. A Reachin Display appears in figure 5.4.

Figure 5.4. The Reachin Display
Courtesy of Reachin Technologies

2.3 Networked haptics

One of the primary concerns with haptic feedback is the device update rate necessary for realistic feeling. While computer graphics images only need to be refreshed roughly 30 times a second (30 Hz) to appear smooth to the human eye, the sense of touch feels vibrations up to 1000 Hz [17]Thus haptic devices must have a control loop that updates nearly 1000 times a second to prevent the user from sensing unwanted vibration.

This order of magnitude difference between visual and haptic refresh rates limits the number and complexity of the models haptic displays can currently handle.

Most haptic devices are run in a separate haptic process or thread, which runs at roughly 1000 Hz. To ensure these high refresh rates, the haptic process may be run on a separate processor or computer [24]. This is typically referred to as networked haptics, and it is an area of ongoing research. Networked haptics are especially important in software such as the IVDA, where a cluster of computing nodes drives the application. Due to the computational stress of a haptic simulation, and the desire to load balance the cluster, it makes sense to have a separate computer devoted to the haptic device. Networked haptics is thus an important consideration for this work.

Kim and Vance demonstrate a method whereby a haptic device's control is executed on a separate haptics computer, which is networked to the computer(s) driving a virtual assembly simulation. The goal is to develop distributed networked haptic environment where multiple users with different haptic devices may work collaboratively in the same virtual environment [25].

The effects of network lag on a haptics simulation are important, especially for collaborative applications. Boukerche et al analyze the effects of lag on simulation performance and present a predictor algorithm to minimize said effects [26]. Similarly, Hikichi et al demonstrate the sensitivity of haptic applications to network packet delay and/or loss [27]. They use prediction and interpolation to smooth over such errors, and then gradually correct back to the proper values as they arrive. Such predictor corrector algorithms are shown to be an effective aid in maintaining a smooth simulation when using networked haptics.

2.4 The Immersive Virtual Design Application

The IVDA is a C++ program written to use the VR Juggler software library developed at Iowa State University's Virtual Reality Applications Center. This permits the application to run on a number of virtual reality devices including the Linux workstation cluster currently being used [28].

Originally, this work used simple linear Taylor series approximations based on pre-computed stress sensitivities to allow a user to change the shape of part and see the stress patterns change interactively [29]. This process was limited by the low accuracy of the Taylor series stress approximations for large design changes and the need to perform a full stress and sensitivity analysis before any sort of interaction could begin. The method also required the portion of the model marked for shape change to be identified beforehand, limiting designer freedom.

The next stage was applying these techniques to a practical engineering problem in a projection screen virtual environment. In particular a tractor rear lift arm experienced excessively high stress levels while in use, but designers found it difficult to alter the shape without interfering with the rest of the complicated lift assembly. The virtual environment with real time stress approximations made it easy to explore the arm design and find a shape that lowered part stress to acceptable levels while avoiding interference with the assembly [30].

Further improvements were provided by Chipperfield, Yeh and Vance with a mesh-free method and a fast reanalysis technique. A reproducing kernel mesh-free method with strain smoothing stabilization was implemented to compute the analysis results [31]. This helped reduce analysis errors arising from mesh distortion as the part shape is changed, and avoids the computationally expensive re-meshing process. The fast reanalysis uses a pre-conditioned conjugate gradient (PCG) method to rapidly resolve the system of equations arising from the mesh-free analysis. By using the factored stiffness matrix from the previous analysis, the PCG method can quickly solve the system for the deformed part shape [32].

In order for a user to arbitrarily choose what portions of the model geometry to change, a form of Free form deformation base on Catmull-Clark subdivision volumes were implemented [33]. By placing a series of control points in the 3D space surrounding the part, the designer creates a subdivided volume that embeds the model. A designer in VR need only grab and move these bounding volume control points to change the underlying part shape.

To make the application applicable to a wider range of problems, Fischer and Vance integrated an external analysis program called Tahoe to perform the mesh-free analyses. Tahoe is a "research-oriented, open source platform for the development of numerical methods and material models" that places special emphasis on solving problems not treated well by standard continuum methods [34]. After testing to compare speed and accuracy with the already implemented custom mesh-free analysis, a Tahoe module was built for the IVDA that makes it the default analysis option.

Stress sensitivity calculation is an important part of the IVDA, since it provides the basis for the Taylor series stress approximations, and it can take up a substantial amount of computation time to produce. To improve both the speed and accuracy, alternative methods were compared from design sensitivity analysis literature to fine a suitable replacement for the finite differences technique being used. A discrete derivatives method with exact numerical differentiation was implemented and shown to be a great improvement, especially for larger models. A photograph of a user within the IVDA appears in figure 5.5 below.

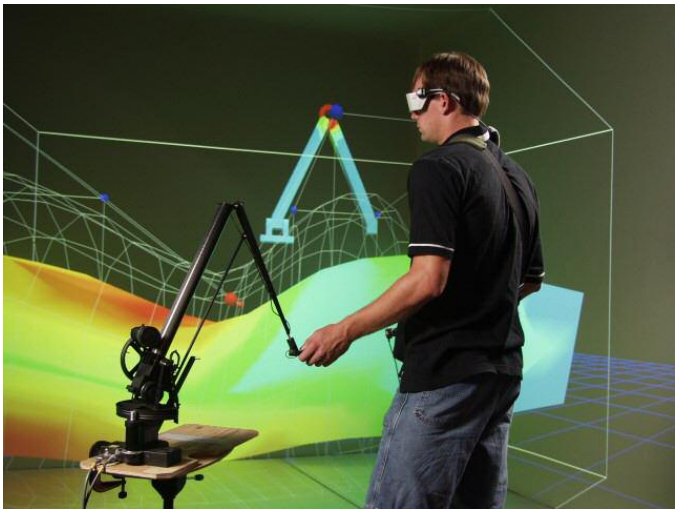

Figure 5.5. A user working with the Immersive Virtual Design Application

With the IVDA in the state outlined above, haptic feedback was added to provide additional information to a designer working with the application in the virtual environment. Decisions were made based on the information obtained from the haptics

literatures and the design and intended use of the application. The process is detailed in the next section.

3. Implementing haptic feedback

Selecting a haptics device was the first step. The three types of devices discussed above were all considered candidates since they are commercially available, they are actively supported, and they come with a documented software API for programming.

3.1 Device selection

The primary requirement for a device with the IVDA is the intended use in a projection screen virtual environment. The difference in physical workspaces needs to be addressed, since the above haptic devices operate on the order of a cubic foot of space while the virtual environment is a 10x10x10ft cube. A method for doing just this is presented in [35], which maps the haptic device workspace to an arbitrary portion of the virtual environment defined by the operator. Still, the larger a haptic device's workspace the more intuitive it is expected to be for a user in the environment.

Another factor to consider is the "strength" of the device; how much force it is capable of simulating for the user. A stronger device provides a potentially wider range of feedback for the user, up to safe limits of course. Device stiffness is an indicator of how rigid the device resistance will feel to the user. More rigid is considered more desirable, as it makes for a more believable simulation.

It also makes sense to avoid 6 degree of freedom haptic devices when the less expensive and less complex 3 degree of freedom devices will suffice. The rotational degrees of feedback would be wasted as the IVDA only supports the translation of bounding volume control points to deform model geometry. This also effectively removes the CyberForce device from consideration, as it is similar to one of the large PHANTOM models, but with the extra glove attachment that is not expected to be useful when working with the IVDA.

The two devices selected for comparison were the largest and strongest offered from SensAble and Force Dimension. Their rankings appear below in Table 5.1.

Device	Max Force(N)	Continuous Force (N)	Stiffness (N/mm)	Workspace Total (m^3)
PHANTOM 3.0	22	3	1	0.2
Delta Device	>20	20	14.5	0.031

Table 5.1. Comparison of haptic device characteristics

The above comparison shows that while both devices may have similar peak forces, the Delta has a much stronger continuous force. The delta is also much stiffer than the PHANTOM, due largely to its unique design. The PHANTOM, however, possesses a much larger total workspace, which became the primary factor for selection. Additionally, the authors have previous experience with a smaller PHANTOM device. In light of these factors the PHANTOM 3.0 was selected for implementation.

Choosing a PHANTOM device also presents the unique choice of a haptics software API to work with. Currently two different API's are provided as an option: GHOST and OpenHaptics. This choice is an important one, since they provide different paradigms for interacting with the PHANTOM.

GHOST is the original PHANTOM software. It a series of C++ classes that provide a scene-graph structure for working with a haptic scene, and is designed for developers with an advanced knowledge of haptics programming. OpenHaptics is the more recent API. It is patterned after the OpenGL graphics language, and applies similar concepts to haptics programming. It was designed to be easy to learn, as well as providing low level device control for advanced developers.

For this work, the OpenHaptics API was selected. It provides a lot of flexibility, and has seen rapid growth in use since it's introduction. It is also more aggressively

pushed by SensAble, including academic downloads of the software, which would indicate it to be the preferred and possibly more likely to be supported API in the long run.

3.2 Integration with the IVDA

Adding networked haptic feedback to the immersive application requires an interface for the device to send and receive information. Keeping the amount of data transferred between the main simulation and the haptic computer to a minimum is a big priority, so the decision needs to be made on just what information the haptic simulation needs.

3.2.1 Network traffic between simulations

Though the designer views the complex geometric model in 3D space, he/she only interacts with that model directly through the placement and manipulation of the bounding volume control points. The model itself is physically transparent to the user. This means the haptic simulation only needs to be aware of the bounding volume control points for manipulation and the bounding volume defining the haptic workspace in the virtual environment.

To produce a feedback for the user the haptic simulation needs an awareness of the stresses (or some other parameter) within the model. However, the haptic device itself is only capable of 3 degree of freedom feedback. This means the most information the haptic device can display at any single instant is a 3 dimensional vector, produced by some algorithm that takes into account the state of the model, which could be run on the immersive application side.

This short analysis indicates the only information we need to share over the network with the device is a series of 3 dimensional vectors for the positions of the various control points and the feedback driving the haptic device. Also note the positions of all control points only need to be sent once when the haptic device is started. While operational only the change in control point positions for the manipulated points needs to

be shared. Network traffic can thus be kept to a minimum, and an interface between the simulations may be designed.

3.2.2 Additions

The immersive application is a collection of C++ modules that interact with each other via specific methods. The core uses VR Juggler to drive the simulation in the virtual environment. Other modules deal with model data, file I/O, free-form deformation, mesh-free analysis, design sensitivity analysis, and Taylor series approximation for stresses. This design lets modules be swapped in an out, such as using Tahoe for the mesh-free and design sensitivity analysis in place of the custom mesh-free implementation.

Interfacing with the haptic computer was done by adding a specialized haptic controller module. This module communicates via TCP/IP data transfer with the haptic simulation computer. The haptic controller sets up the simulation by formatting and sending the locations of all control points from the model manager to the haptic computer. It then monitors for updated control point positions and passes those positions back to the core application to cause deformations. As the model shape changes, the haptic controller uses an algorithm to convert the changing stress state into a value to return to the haptic device for feedback.

The haptic controller is also an optional component. If no haptics are used, it does not need to be loaded or even compiled into the application. Figure 5.6 below shows a diagram of the IVDA application and the haptic controller module.

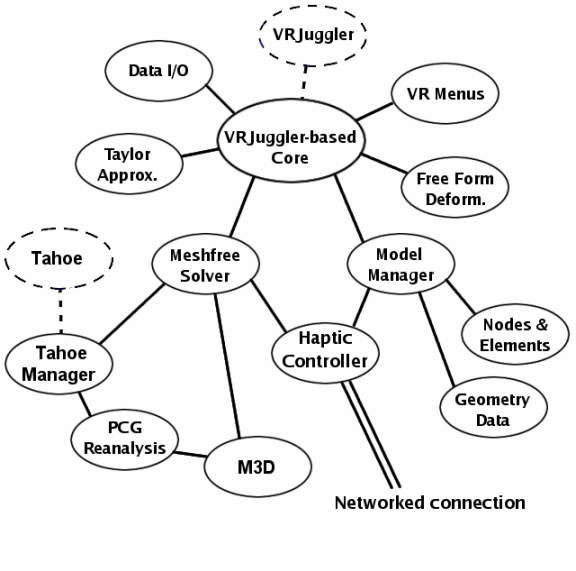

Figure 5.6: The haptic controller module in the IVDA

3.3 The haptic server

On the dedicated haptic computer, a program called the haptic server was written to control a haptic device and communicate with the IVDA simulation. This lets the haptic computer dedicate itself to driving the haptics and keeping update rates high. The haptic was written using the OpenHaptics API to control the PHANTOM 3.0.

The haptic server is actually started before the IVDA simulation. It simply waits for a connection from a client, the haptic controller. Once connected it receives data and parses it for the PHANTOM, setting up control points and workspace bounds. When the haptic simulation starts, the haptic server relays control point translations to the IVDA and receives the user feedback information. This feedback is converted into a parameter meaningful to the PHANTOM.

Since the haptic server only updates the feedback when new information is received from the simulation, the PHANTOM servo loop may run as fast as possible. If the change in feedback levels substantial, a linear interpolation may be used to smoothly apply the new forces to the haptic device, as in [27]. A graphical representation of the haptic server communicating with the IVDA's haptic controller and the PHANTOM appears in figure 5.7.

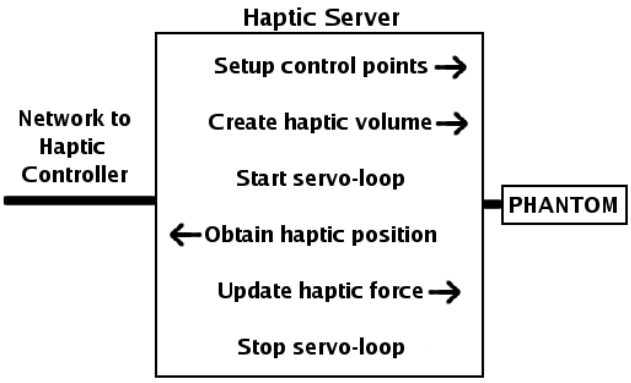

Figure 5.7. The haptic server

4. Model state to haptic feedback correlation

The described work permits the IVDA to work with a PHANTOM haptic device for feedback based on the changing stress state of the model being altered. There are two missing steps to this process: 1. Deciding just how to actually convert model stress information into a single scalar or 3 dimensional vector for the haptic feedback. 2. Finding a way to apply this feedback in a way that is meaningful to the user.

4.1 Model stress conversion

Converting the model's changing stress pattern into a value for the haptic device is a highly empirical process. The goal is to produce a "feeling" for the user that conveys as much information about the stress state as possible, so a large amount of testing is expected.

Each mesh-free node (or "element") has a stress tensor associated with it, and a model typically has thousands of elements. The IVDA allows the user select one of several different stress states to view, such as Von-Mises or maximum shear. This provides a scalar value for each element.

The next step is to come up with a sort of "average" value to send to the haptic device. The most straightforward technique is probably the global mean of all stresses in the model as per equation 1.

$$\gamma = \sum_{i=1}^{N} \sigma_{VM}(i) \qquad \textbf{eq. 1}$$

Here γ is the haptic feedback value (here simply the mean stress), σ_{VM} the Von-Mises stress per element, and N the number of elements in the model. The disadvantage of this approach is the expected insensitivity to design changes, especially when smaller portions of the model are being changed.

A weighted mean is another possibility, where the stress values are weighted in some way by the stress sensitivities used in the Taylor series approximation. This should cause the areas being deformed to have a greater effect on the haptic feedback. This also removes the effect of stresses in areas not being deformed from the haptic feedback. Such a weighting is shown in equation 2.

$$\gamma = \sum_{i=1}^{N} \bar{h}(i)\sigma_{VM}(i) \qquad \textbf{eq. 2}$$

Here $\bar{h}$ is the average value of the stress sensitivities for each element, computed for the three coordinate directions x, y and z.

$$\bar{h} = \frac{h_x + h_y + h_z}{3} \qquad \textbf{eq. 3}$$

A third option is to provide a different level of haptic feedback for each coordinate direction. This might be accomplished in a manner similar equation 2, except we would have a weighted mean for each coordinate direction. This would also require returning a feedback vector to the haptic device instead of a scalar.

The last step before sending feedback to the device is to map it to some sort of range or scale. A value for haptic feedback needs to be seen within the context of a maximum and minimum to be meaningful. The minimum and maximum values used depend on the averaging method. For the first, the simple global average, the minimum

and maximum stresses in the model is used. For the second (and third) the minimum and maximum of the stress times sensitivity value is used.

Finally, the feedback value(s) are mapped from 0 to 1 for convenience, where 0 is no feedback and 1 is the maximum feedback the haptic device is programmed to provide. Equation 4 presents this mapping, where γ_{haptic} is ranges from 0 to 1 and $\gamma_{\min}$, $\gamma_{\max}$ are the minimum and maximum as determined above.

$$\gamma_{haptic} = \frac{\gamma - \gamma_{\min}}{\gamma_{\max} - \gamma_{\min}} \qquad \textbf{eq. 4}$$

Note that if we are using different feedback for each of the 3 coordinate directions, γ_{haptic} becomes a 3 dimensional vector instead of a scalar.

4.2 Haptic feedback representation

Once a value is generated to send to the haptic device, the second step is deciding just how to provide the haptic feedback on the device. At a higher level, haptic devices are typically used to provide two broad types of feedback: force/resistive feedback and vibration or tactile feedback.

Earlier, the PHANTOM 3.0 was selected for this work. It is designed mainly to produce whole-arm force feedback, so that type was selected. Ideally this will let the haptic device "guide" the user to a more optimal solution by making motion is some directions easier than in others.

Next, a method of applying the force/resistive feedback needs to be determined. The Open Haptics toolkit provides some examples of force feedback models that were considered. Consider the standard mass-spring-damper system, which appears in equation 5.

$$m\frac{\partial^2 x}{\partial t^2} + b\frac{\partial x}{\partial t} + kx = F \qquad \textbf{eq. 5}$$

Here m is the mass, b the damping constant, and k the spring constant. The feedback value could be used to adjust any one of these parameters.

Practically, the mass m was discarded altogether because it would depend on the acceleration, or how rapidly the user changed moving the PHANTOM about. It gives the feeling of a weight being attached to the end of the device. This leaves the damping constant and the spring constant.

Adjusting the damping constant b generates a viscous, friction like effect to resist the motion of the user. As γ_{haptic} ranges from 0 to 1, b is ranged from 0 to a some maximum value determined by experiment with the device. Too high of a value produces forces large enough to shut the PHANTOM down.

The stiffness constant k gives a direct spring force to resist motion based on the position of the device. Varying this value changes how difficult it is for the user to move the device, and hence deform the model. A suitable range of k is also determined by experiment with the device to avoid excessive force levels.

With no clear way to prefer one force rendering method over another, both choices were implemented in a small pilot study designed to help determine how useful haptic feedback is to a user working with the IVDA. Details of the study are presented in section 5.

5. Pilot study

A small pilot study was setup and run with a two-fold goal: 1. determine if a user of the IVDA perceives any benefit from force feedback tied to the stress levels in the deforming models and 2. see if there is a user preference between spring force feedback and friction or damping feedback. This was designed as a precursor to a larger scale user study on the effectiveness of the various stress to haptic feedback mapping techniques. For this pilot study, the simple stress averaging technique was used.

5.1. Setup

A total of eleven users participated in the study, each with varying levels of computer usage experience. Video game use was questioned as well, since that was

expected to have an impact on the user's willingness to experiment with the haptic device. Each participant was given a pre-study and a post-study questionnaire.

Users had the application explained to them, they completed the pre-study questionnaire, and were then placed in front of a sample version of the IVDA. This simplified version had a simple beam model already loaded with a bounding volume defined, control points selected for deformation, and stress sensitivities computed. The tests were performed with a desktop VR setup instead of the intended immersive display for simplicity. Users wore goggles to provide stereo vision.

Participants were asked to deform the model with the haptic device for both spring and friction feedback. Users also had the chance to work with two different haptic devices, a small PHANTOM Omni and the larger PHANTOM 3.0 which the application was designed for. The post-study questionnaire was then completed.

5.2 Results

The first result of interest from the study was the type of feedback users preferred. The overwhelming choice was the spring force. The friction force was rejected by all. One user thought the force interfered with deforming the model, and another had no preference. A simple graph of these results appear in figure 5.8.

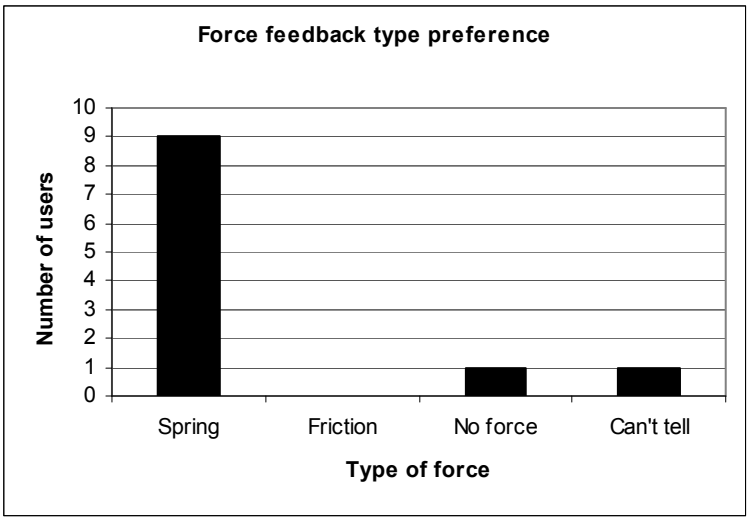

Figure 5.8. Comparing the different types of force feedback

The second result was to determine a preference for haptic devices. It was expected the 3.0 would be preferred over the Omni due to it's larger strength and size. This was indicated by the study as well as figure 5.9 shows.

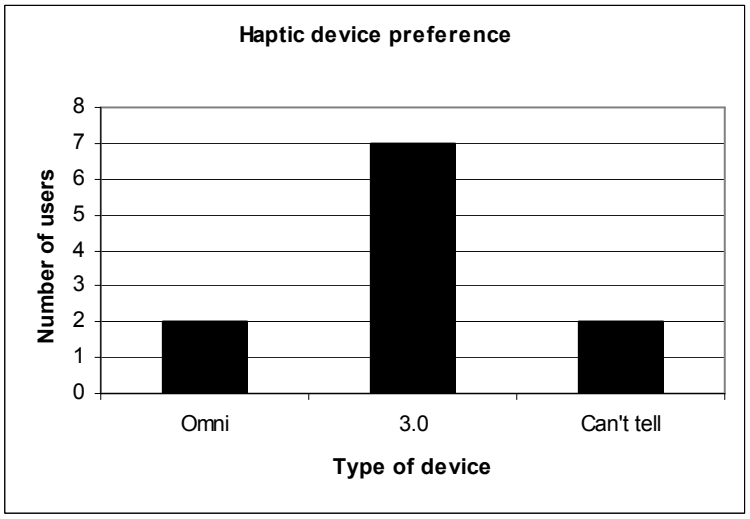

Figure 5.9. Comparing two different PHANTOM devices

This simple study was helpful in reinforcing the choice of the PHANTOM 3.0, and in deciding to use the spring force feedback model for further work with the IVDA.

6. Conclusions and future work

Haptic feedback based on the changing stress levels in a deforming model was added to the Immersive Virtual Design Application. An overview of haptic devices and uses was presented, with an emphasis on networked haptics. Haptic feedback was implemented with the existing IVDA and several mappings from model stress state to haptic feedback developed. A simple pilot study was run to evaluate the different types of haptic feedback available to users of the IVDA to determine the most effective.

Future work would focus on the implementation of a larger scale user study to compare a number of different stress mappings for haptic feedback. Combined with the PHANTOM 3.0 and spring force feedback, this future study should be used to determine the most useful way to couple haptic feedback with the immersive design application.

7. Acknowledgements

The authors would like the thank the participants of the pilot study for their time, the summer research program participants, and the Iowa State University Virtual Reality Applications Center.

8. References

1. Gupta, R., T. Sheridan, and D. Whitney, *Experiments Using Multimodal Virtual Environments in Design for Assembly Analysis*. Presence, 1997. 6(3): p. 318-338.
2. Avila, R.S. and L.M. Sobierajski. *A Haptic Interaction Method for Volume Visualization*. in *Visualization '96*. 1996. San Francisco, CA: IEEE.
3. Pausch, R., D. Proffitt, and G. William. *Quantifying Immersion in Virtual Reality*. in *24th annual conference on Computer graphics and interactive techniques*. 1997: ACM Press.
4. Burdea, G.C., *Force and Touch Feedback for Virtual Reality*. 1996, New York: John Wiley & Sons, INC.
5. Burdea, G.C., *Invited Review: The Synergy Between Virtual Reality and Robotics.* IEEE Transactions of Robotics an Automation, 1999. 15(3): p. 400-410.

6. Volkov, S. and J. Vance, *Effectiveness of Haptic Sensation for the Evaluation of Virtual Prototypes.* Journal of Computing and Information Science in Engineering, 2001. 1.
7. Perkowitz, S., *Feeling is Believing*, in *New Scientist*. 1999. p. 34-7.
8. Massie, T. and J.K. Salisbury. *The PHANTOM haptic Interface: A Device for Probing Virtual Objects*. in *ASME Symposium on Haptic Interfaces for Virtual Environments*. 1994. Chicago, IL: ASME.
9. SensAble Technologies, *Haptic Devices*. Sensable Products and Services website. 2006. http://www.sensable.com/products-haptic-devices.htm. July 30, 2006.
10. SensAble Technologies, *The GHOST SDK Programmer's Guide*. 2001. Included with the GHOST SDK software.
11. SensAble Technologies, *OpenHaptics Toolkit Programmer's Guide*. 2005. Included with the OpenHaptics software.
12. Immersion Corporation, *CyberGrasp Exoskeleton*. 3D Interaction website. 2006. http://www.immersion.com/3d/products/cyber_grasp.php. July 30, 2006.
13. Immersion Corporation, *CyberForce Tactile Feedback System*. 3D Interaction website. 2006. http://www.immersion.com/3d/products/cyber_force.php. July 30, 2006.
14. Grange, S., et al. *The Delta Haptic Device*. in *Mecatronics 2001*. 2001. Besancon, France.
15. Force Dimension, *Delta Haptic Device*. Force Dimension products webiste. 2006. http://www.forcedimension.com/fd/avs/home/products/6-dof_delta/index.html. July 20, 2006.
16. SensAble Technologies, *FreeForm Modeling and Modeling Plus Systems*. Sensable Products and Services website. 2006. http://www.sensable.com/products-freeform-systems.htm. July 30, 2006.
17. Salisbury, J.K. and M.A. Srinivasan, *Phantom-Based Haptic Interaction with Virtual Objects.* IEEE Computer Graphics and Applications, 1997. 17(5): p. 6-10.
18. Chen, E. and B. Marcus, *Force Feedback for Surgical Simulation.* Proceedings of the IEEE, 1998. 86(3): p. 524-530.

19. McNeely, B., *Introduction to VPS*. 2002, The Boeing Company. Included with the VPS software.

20. McNeely, W.A., K.D. Puterbaugh, and J.J. Troy. *Six Degree-of-Freedom Haptic Rendering Using Voxel Sampling*. in *26th Annual Conference on Computer Graphics and Interactive Techniques*. 1999.

21. Howard, B. and J.M. Vance, *Desktop haptic virtual assembly using physically-based part modeling.* Virtual Reality, In Review.

22. Harding, C. and M.C. Newcomb, *Supporting Interactive Data Exploration for GIS Planning Tasks with a Multi-modal Virtual Environment.* Third IEEE Workshop on Haptic, Audio and Visual Environments (HAVE) 2004, 2004: p. 81-86.

23. ReachIn Technologies, *The ReachIn Display*. Reachin products website. 2006. http://www.reachin.se/products/Reachindisplay. July 30, 2006.

24. Burdea, G.C. *Haptics Issues in Virtual Environments*. in *CGI 2000*. 2000. Geneva, Switzerland: IEEE.

25. Kim, C. and J.M. Vance. *Development of a networked haptic environment in VR to facilitate collaborative design using Voxmap Pointshell (VPS) software*. in *ASME Design Engineering Technical Conferences and Computer and Information in Engineering Conference*. 2004. Salt Lake City, Utah.

26. Boukerche, A., S. Shirmohammadi, and A. Hossain, *Moderating Simulation Lag in Haptic Virtual Environments.* Proceedings of the 39th Annual Simulation Symposium, 2006: p. 269-277.

27. Hikichi, K., et al. *Architecture of haptics communication system for adaptation to network environments*. in *IEEE International Conference on Multimedia and Expo*. 2001: ICME.

28. Bierbaum, A., et al. *VR Juggler: A Virtual Platform for Virtual Reality Application Development*. in *Virtual Reality, 2001*. 2001. Yokohama, Japan: IEEE.

29. Yeh, T.-P. and J. Vance, *Applying Virtual Reality Techniques to Sensitivity-Based Structural Shape Design.* ASME Journal of Mechanical Design, 1998. 120(4): p. 612-619.

30. Ryken, M.J. and J.M. Vance, *Applying Virtual Reality Techniques to the Interactive Stress Analysis of a Tractor Lift Arm.* Finite Elements in Analysis and Design, 2000. 35: p. 141-155.

31. Chen, J.S., et al., *A stabilized conforming notal integration for Galerkin mesh-free methods.* International Journal for Numerical Methods in Engineering, 2001. 50: p. 435-466.

32. Chipperfield, K., J.M. Vance, and A. Fischer, *Fast Meshless Reanalysis Using Combined Approximations, Pre-Conditioned Conjugate Gradient, and Taylor Series.* AIAA Journal, In Review.

33. MacKracken and Joy. *Free-Form Deformations With Lattices of Arbitrary Topology*. in *SIGGRAPH 96*. 1996.

34. Sandia National Labs, *Tahoe Development Server*. Sandia National Labs website. 2005. http://tahoe.ca.sandia.gov. July 11, 2006.

35. Fischer, A. and J.M. Vance. *PHANToM Haptic Device Implemented in a Projection Screen Virtual Environment*. in *7th International Immersive Projection Technologies Workshop*. 2003. Zurich, Switzerland.

CHAPTER 6. SOFTWARE ENGINEERING FOR INTERACTIVE DESIGN

Working with the interactive virtual design application as a research project necessarily involves a lot of software work, both in design and programming. One needs to keep the goals of the IVDA in mind when improving and extending it's functionality, namely that it remain fast and modular. This section discusses the software engineering aspects of the work described so far.

The application and all of the major external software it uses, VR Juggler, OPCODE, SuperLU and Tahoe, are written in C/C++, mid-level programming languages that provide both speed and flexibility. However, this power comes at a price, as these languages don't provide garbage collection or much protection from programming and memory management mistakes. Keeping track of memory is extra important in an analysis application, since larger models can use up a good portion of the available system memory; memory leaks will quickly kill such a simulation.

1. Simulation speed

A key feature when working in Virtual Reality is that the simulation maintain real-time update rates. As stated, that is typically assumed to be around ($1/15^{th}$) of a second, since below that level the simulation generally seems smooth to the human eye. Any time the update rate gets slower, the user will notice and the immersive effect will degrade. This update effect not only applies to the visual update rate, but to the user's control as well. If a model being moved by the user doesn't update its position quickly enough, it will drastically affect the experience.

This means any action that must take place in real-time, such as deforming a model and seeing the stress pattern update, has to be fast. That is why the linear Taylor

Series stress approximation is used even though it's poor for anything other than small design changes. Speed here becomes more important than accuracy, as the project's goal is to help designers gain an intuitive feel for how their shape changes affect the analysis results. If more accurate results are desired, they should be analyzed for in a separate thread.

Threading, the running of multiple execution processes, is an essential part of the IVDA. One of the worst experiences for a user in VR is to have the simulation freeze. They lose all sense of immersion, and either become frustrated or quickly assume the application has failed. Any task that requires a substantial amount of computation, such as a full mesh-free analysis, stress sensitivity calculation, or even loading a complex model, must be launched in a separate thread so the application continues to respond to user input.

Fortunately VR Juggler provides a built-in interface, VPR, that helps make threading more straightforward. The designer must ensure that the application itself is thread-safe, so that multiple functions don't attempt to modify the same data at the same time. It also pays to be aware of the types of threads created. On multiprocessor systems, such as SGI workstations, certain threading models may automatically spawn on different processors, which helps load balance the application. Others will always thread on the same processor.

2. Distributed computing

Currently, running an application in a large projection screen virtual environment usually means it is using one of two types of systems:

1. A large, multi-graphics pipe, shared memory parallel computer
2. A distributed cluster of high end PC workstations

Both present an opportunity for parallel processing, where a computationally intensive task is split up among multiple processors for a faster solution time and more efficient use of computational resources.

The large shared memory computer is becoming less common today due to the increasing power of PC graphics cards and their much lower cost. This means any parallel computation done with the IVDA is likely to be the distributed memory type, where each computer (or node) has it's own copy of any data. Distributing this data well is important for an efficient parallel calculation.

VR Juggler has methods for sharing information across a cluster with the ApplicationData interface. It provides a low-level way to divide data up among the different nodes. At a higher level, the Message Passing Interface, MPI is a software library used to send information across computing clusters. The IVDA can use MPI in the analysis matrix assemble and solve steps if desired. Additionally, both Tahoe and SuperLU permit distributed parallel processing for a number of routines, especially when solving systems of equations, all using MPI for the actual data distribution.

However, distributed parallel processing should not be blindly applied in every case. Often when using the IVDA, the models under consideration are small, ranging from several hundred to a few thousand elements. In those cases the additional time required to distribute the data and gather the solution across the network can outweigh, or at least minimize, any speed benefits gained from parallel processing. This is especially true when the cluster is connected with a lower speed interconnect such as 100BaseT networking, as may often be the case with VR clusters.

Lastly, it pays to remember that in a typical projection screen VR environment the simulation cluster and the computation cluster (say for solving large finite element problems) are one and the same. Even when parallel processing each node is still running

the virtual environment at the same time. Keeping the user immersed in the virtual environment should be the most important task.

3. Optimization

Optimization of computer code, as opposed to optimization of mechanical design problems, is another important step. Several software tools were used with the IVDA to help profile the application to determine which routines were the most computationally expensive.

On SGI systems, the Workbench suite of tools provides an in-depth profiler that was used to determine the slowest portions of the stiffness matrix assembly routines. This information helped speed the assembly process up by a factor of two in the original mesh-free solver. When working with Linux, as is typically found on cluster VR systems, the GNU gprof profiling tool is used. Gprof generates several pieces of information including a "flat profile" of how much time was spent in each function, including the number of time it was called, and a "call graph" of which functions called which. This information helped optimize the discrete derivative calculation within Tahoe.

Less fancy, but no less important, than profiling is keeping some basic rules of efficient computing in mind. When possible, arrays should be accessed as they are contiguous in memory. This can provide a substantial performance increase, and is especially important when doing many linear algebra operations, such as those found in the IVDA. It is also wise to pre-compute and store as much information as possible. In mesh-free methods, a substantial amount of data may be stored and used later in this fashion.

Lastly, there is the issue of data translation. Trying to make the immersive design application generally useful means being able to work with geometry from a variety of sources. Model format converters were written to move between ABAQUS input, Tahoe

data, EXODUS, and the mesh-free model file formats. These converters also make it possible to reanalyze deformed models reached through using the IVDA with other analysis software.

CHAPTER 7. SUMMARY AND FUTURE WORK

This research demonstrated the use of virtual reality and related interaction technology, such as haptics, in an immersive virtual design application. The goal was to present an effective methodology for interactive design where engineers alter products and investigate updated analysis results in real-time using a virtual environment. Research was presented as a series of four papers.

Chapter 2 covered a history of the interactive design methodology and it's applications followed by the largest hurdles to effective interaction. Solutions where presented and their effectiveness demonstrated.

The third chapter examined some limitations of the custom mesh-free analysis method used on the application. A search was performed for suitable replacement software with an emphasis on Open Source applications. The Tahoe program was selected, tested against the custom implementation, and integrated with the application.

In Chapter 4 the calculation of shape design sensitivities for stress approximation were examined. Limitations of the existing finite differences technique were explained and a search of available techniques was made. After considering discrete derivatives, continuum derivatives, and automatic differentiation, and discrete derivatives approach with exact numerical differentiation was selected, implemented, and tested. Results indicate the discrete derivatives approach to be a fine replacement.

Chapter 5 considered the use of haptic or force feedback as an additional channel of information for the designer. Several haptic devices and their uses were examined, and a PHANTOM 3.0 was selected for use with the immersive design application. Networked haptic feedback was added to the application and several techniques for mapping model stress state to device forces were investigated. Finally, a simple pilot study was conducted to help determine which feedback styles were considered most useful to a potential designer.

The area with the greatest potential for future work would seem to be the use of haptics as a design tool in the application. A proper user study should be conducted to sort out which stress-force mapping techniques are the most effective for a designer. One

should also try to determine just how useful the additional haptic feedback generally is when working in the virtual environment.

A second area worth investigating further is the use of automatic differentiation routines for the computation of shape design sensitivities. Throughout the course of this research automatic differentiation has made many improvements. With programs as large as 400,000 lines and more now being completely differentiated, it appears to be fast becoming a practical analysis option. This would also make implementing additional types of sensitivity analyses much more straightforward, and would be a powerful addition to the methodologies of interactive design in virtual reality.