An approach to convert vertex-based 3D representations to combinatorial B-splines for real-time visual collaboration

Vijay Kalivarapu
Iowa State University, vkk2@iastate.edu

Eliot H. Winer
Iowa State University, ewiner@iastate.edu

An approach to convert vertex-based 3D representations to combinatorial B-splines for real-time visual collaboration

Abstract
Scientific Visualization and Virtual Reality are increasingly being used for the design of complex systems. These technologies offer powerful capabilities to make decisions that are cost and time effective. The next logical extension is to collaborate with these visual models in real-time, where parts of a design team are geographically separated. Specifically, visual collaboration enables ideas and proposed changes to be discussed exactly on a virtual model of a product. However, high-end visualization hardware and Internet technologies impede widespread use of real-time visual collaboration due to the large amount of data from which these representations are created. These data are typically in the form of 3D vertex-based models, which offer a high degree of realism when displayed, but at a price of storage, rendering speeds and processing efficiency. The more realistic the representation desired, the larger the number of vertices required and hence the higher the file size. In this paper, we propose a new data modeling and handling technique where traditional vertex-based models are converted into combinatorial B-Spline based wire-frame models that allow realtime visual collaboration in the context of typical virtual reality systems. Using appropriate filtering methods, parametric equations are computed for each curved segment in a vertexbased representation and bundled together with sampled linear segments of the model. The computed parametric equation based models occupy only a fraction of the size when compared to the original vertex-based models. These lightweight models can easily be transmitted over the Internet, in real-time, for viewing with a platform independent visual client program. The proposed methods were tested on several example data files to prove the method's effectiveness.

Keywords
Virtual Reality Applications Center, data modeling, vertex-based models, virtual reality

Disciplines
Computer-Aided Engineering and Design | Mechanical Engineering

Comments
This is a conference proceeding from *43rd AIAA Aerospace Sciences Meeting and Exhibit - Meeting Papers* (2005): AIAA 2005-124, doi: 10.2514/6.2005-124. Posted with permission.

43rd AIAA Aerospace Sciences Meeting and Exhibit
10 - 13 January 2005, Reno, Nevada

AIAA 2005-124

An Approach to Convert Vertex-Based 3D Representations to Combinatorial B-Splines for Real-Time Visual Collaboration

Vijay K. Kalivarapu[*] and Eliot H. Winer.[†]
Department of Mechanical Engineering
Iowa State University, Ames, IA, 50011

Virtual Reality Applications Center
Iowa State University, Ames, IA, 50011

Scientific Visualization and Virtual Reality are increasingly being used for the design of complex systems. These technologies offer powerful capabilities to make decisions that are cost and time effective. The next logical extension is to collaborate with these visual models in real-time, where parts of a design team are geographically separated. Specifically, visual collaboration enables ideas and proposed changes to be discussed exactly on a virtual model of a product. However, high-end visualization hardware and Internet technologies impede widespread use of real-time visual collaboration due to the large amount of data from which these representations are created. These data are typically in the form of 3D vertex-based models, which offer a high degree of realism when displayed, but at a price of storage, rendering speeds and processing efficiency. The more realistic the representation desired, the larger the number of vertices required and hence the higher the file size. In this paper, we propose a new data modeling and handling technique where traditional vertex-based models are converted into combinatorial B-Spline based wire-frame models that allow real-time visual collaboration in the context of typical virtual reality systems. Using appropriate filtering methods, parametric equations are computed for each curved segment in a vertex-based representation and bundled together with sampled linear segments of the model. The computed parametric equation based models occupy only a fraction of the size when compared to the original vertex-based models. These lightweight models can easily be transmitted over the Internet, in real-time, for viewing with a platform independent visual client program. The proposed methods were tested on several example data files to prove the method's effectiveness.

Nomenclature

t	=	Normalized parameter varying between 0 and 1
n	=	Number of control points
B_i	=	i^{th} control point
$N_{i,k}(t)$	=	Cox and de Boor's recursive Basis functions
u_i	=	Knot vector
k	=	Order of B-Spline curve

I. Introduction

Scientific visualization and Virtual Reality (VR) are increasingly becoming key tools in complex design processes for engineered parts. Interpreting three-dimensional representations instead of numbers and text

[*] Research Assistant, Mechanical Engineering Department, 2274 Howe Hall, Room 1620, Student Member.
[†] Assistant Professor, Mechanical Engineering Department, 2030 Black Engineering Building, Member

enormously increases the understanding of complex data. Visualization transforms data into graphical representations that exploit the high-bandwidth channel of the human visual system, leveraging the brain's remarkable ability to detect patterns and draw inferences [1].

Three-dimensional visualization has enabled engineering design to attain new heights of accuracy, efficiency, and reliability. However, further performance improvements in the design of complex products or processes are always desired, particularly in the commercial marketplace. Thus, the ability to see larger amounts of design data, and in new ways, will always remain a requirement. New issues arise and must be dealt with, as the use of high-end visual environments and tools increases. One of these issues is collaborating or sharing a visual representation. Most virtual parts start as models from Computer Aided Design (CAD) (i.e. surface and/or solid modeling) software. However, there exists difficulty with getting a native CAD model into a VR environment in an efficient manner. VR systems rarely use CAD models directly and some format conversion must take place. This often results in the models becoming a cache of points (vertices), whereas originally they may have been a different model format. Most CAD packages implicitly utilize parametric equations for representing curves and surfaces. These exact forms are specific to each vendor and are often proprietary. To use these representations for other purposes (i.e. VR, structural analysis), the models must be exported, and are usually not in parametric form. Typically these exported 3D representations contain a framework of vertices sewed together with graphical algorithms to form a surface or solid model. Implicitly, the more complex the part or product being represented, the more vertices required to accurately display it and the larger the size of the data source. Current advancements in computational technologies allow designers to interact with much of this data directly if they are collocated with it.

It is common in today's marketplace for a company to have a design process occurring in multiple locations simultaneously. Thus, the issue of collaboratively working on a design becomes critical. However, in a collaborative design environment[2, 3], where designers are geographically distributed, formats, interfaces, and data sizes become obstacles to effective design collaboration. Collaborative visualization (CV) offers capabilities to have faster design cycles with better results, but currently there is a limit to the amount of data that can be traversed between distributed design teams. Data exchanges between teams in a collaborative VR environment often take place over the Internet. Today's Internet connectivity speeds are constrained by many factors including existing network hardware (i.e. router speeds), available bandwidth, and latency. Thus, the maximum speed that is truly attainable is far less than the requirements of designers in a collaborative scenario. Complex engineering design can easily produce files containing millions of vertices on the order of gigabytes or terabytes in size. In addition, as engineers may be using different computer platforms and operating systems, type commonality, and availability of the user interface (UI) arise. Even if sending a vertex-based model over the Internet in real-time was always possible, the question arises of what a user can do with it. Fast data transfer is not enough; capabilities for visualizing and interacting with the representations must be developed. Thus, there is a need for theories that take vertex based representations, process them and make them available to any user in any location in real-time, especially those on high-end VR systems. Paramount to this theory is for the visual representations to be compact, while still maintaining accuracy.

In this paper, the development of a novel theory to produce, transport, and visualize these representations is presented. Vertex based models of engineering parts are transformed into a collection of linear and non-linear curves, which when drawn simultaneously, produce a newly created model of the original part. This method creates these representations at a fraction of the storage of the original model. The theory also provides a mechanism to easily transport these representations over the Internet for real-time viewing on any computing platform.

II. Background

A considerable amount of research has already been done in CV in a virtual environment. Projects and languages such as the Virtual Reality Modeling Language (VRML)[‡], Sieve[4], Collaborative Environment for Visualization Using Java RMI (CEV)[5], Web based Collaborative Visualization of Distributed and Parallel Simulation[6], Distributed Objects Based Scientific Visualization Environment (DOVE)[7] and SGI VizServer[§] are several examples of technologies developed or used for CV. The performance of these environments and software directly depend on Internet connectivity speeds, and as such their real-time performance is subjected to file size limitations. These projects (with the exception of VizServer) have issues with large file sizes and requirements for high-bandwidth, low latency networks. In addition, many of these software implementations require specialized software on a user's computer necessitating learning of a detailed interface or scripting language to properly use the software. SGI

[‡] Web Reference: VRML specifications: http://www.web3d.org/x3d/specifications/vrml/

[§] Web Reference: SGI VizServer: http://www.sgi.com/software/vizserver/

Vizserver was different in that it used standard OpenGL implementations to perform collaborative visualization by transmitting frame buffers rendered at the server over the Internet (i.e. "screen scraping"). However, even with advanced compression techniques such as color-cell compression and interpolated-cell compression with a substantial network connection (100Mbps)[11], only frame rates of 14 – 16 frames-per-second (fps) was attained for models with any complexity. Smooth motion in a visual animation typically requires a frame rate of 25 – 30 fps, and anything lower can greatly reduce visual quality and real-time interactivity.

A second approach to CV was to use existing technologies in CAD systems. Teamcenter[**] from UGS and iSight[††] from Engineous Software are examples of software tools built to allow collaboration between CAD, analysis, and optimization environments. These packages offer tremendous capabilities as they often tie directly into the proprietary representations internal to a CAD system. However, there are significant drawbacks associated with these approaches. First, integration of these systems with high-end visual environments is very complex and often times impossible. Second, each designer must have required software on the computer to be linked, even if for only a small number of collaborative sessions. Lastly, the interfaces for these packages (and similar ones) are often geared towards manipulating files in a shared directory structure. Very little thought has been put into the interface to enable real-time communication and data-sharing that will foster improved design processes.

A final class of methods and tools have centered on altering the data used for the visual representation. Decimation and polygonal simplification techniques[9, 10] have been developed that can significantly reduce file sizes and thus, reduce transfer time. However, these reductions often occur with a significant loss of information and therefore prevent accurate portrayal of the 3D data. They also do not provide a consistent means from computer to computer to view and interact with the representation.

A promising method was proposed by H. Hoppe. This method involves reconstructing the surfaces[8] of a model from the vertex data using triangulation followed by B-Spline interpolation. The main drawback is that this method was intended to improve the visual quality of a model. So, model compactness was not a priority of the theory and use for collaborative visualization is therefore limited. That notwithstanding, it is this premise that the method presented in this paper is built. The hypothesis set forth is that a set of parametric curves and surfaces can be constructed from a set of vertex data at a drastically reduced file size compared to the original representation. These parametric representations would maintain each vertex point, representatively, thus keeping accuracy high. In addition, a web-based server environment will be used to transport the representation through a common web-browser on any computing platform in any geographic location through a simple to use, yet powerful interface enabling true collaborative visualization amongst high-end virtual environments.

III. Method Development - Conversion of Three-Dimensional Vertex Data to combinatorial B-Splines

The vast range of formats, which vertex data can be stored, immensely complicated completion of the proposed theory. As such, the contents of this paper present a method to convert vertex data into parametric curves leading to a wireframe representation of the model under consideration. While this in itself is not enough to handle extremely complex geometries, it is crucial to gain the necessary understanding to formulate the full theory that will, in turn, enable complex representations including surface modeling, shading, and texturing.

These new representaitons are a combination of linear and parametric B-Splines. Parametric equations for B-Spline non-linear segments are computed and the necessary coefficients and parameters are stored along with filtered linear components. These models, when used in a collaborative scenario, can reduce network bandwidth overhead, and facilitate real-time CV between geographically distributed design teams. This research approach is a unique attempt to address real-time CV. All other methods attempt to deal with the full vertex-based data and somehow transmit it over a network or the Internet or a means to reduce it with minimal loss of accruacy. The method proposed here reduces the data to a parametric representation, which enables a large decrease in data storage, with no to a negligible loss of accuracy. While the use of parametric representations is not new in visualization, this means of storing and using the data has never been used in collaborative VR systems. Several file formats such as STEP[‡‡] and IGES[§§] already provide some model representation in parametric form, but this is

[**] UGS: Products: Teamcenter, Web Reference: http://www.ugs.com/products/teamcenter/, 2004.

[††] iSIGHT - Integrate, Automate, and Optimize your Manual Design Processes, Web Reference: http://www.engineous.com/product_iSIGHT.htm, 2004

[‡‡] ISO STEP Standards – STEP Tools, Inc., Web Reference: http://www.steptools.com/library/standard/index.html, 2004.

[§§] Initial Graphics Exchange Specification, IGES Project, Web Reference: http://www.nist.gov/iges/, 2004.

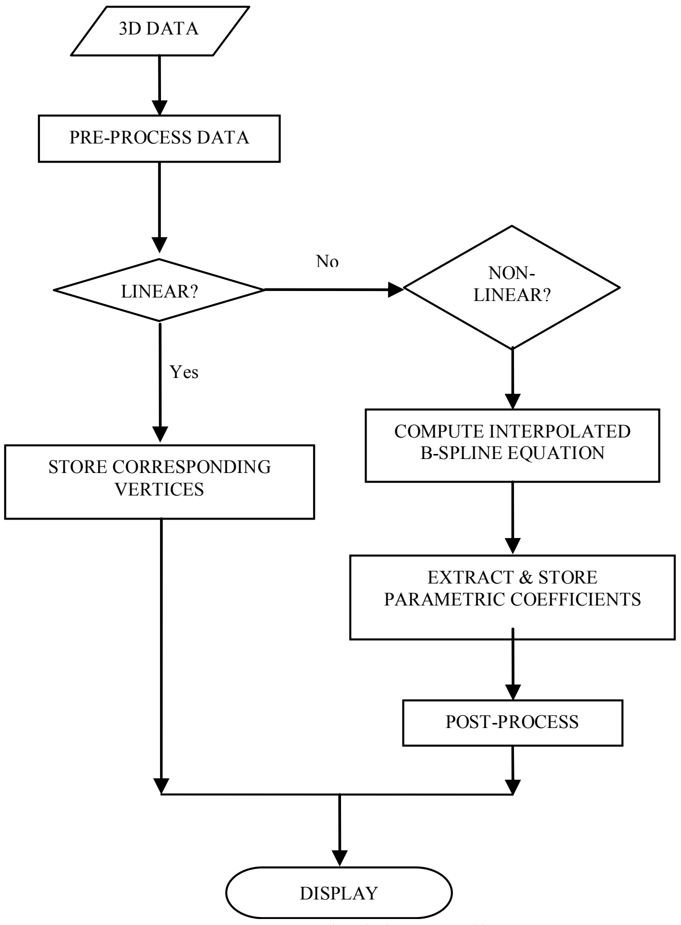
3D DATA
PRE-PROCESS DATA
LINEAR?
No
NON-LINEAR?
Yes
STORE CORRESPONDING VERTICES
COMPUTE INTERPOLATED B-SPLINE EQUATION
EXTRACT & STORE PARAMETRIC COEFFICIENTS
POST-PROCESS
DISPLAY

Figure 1 - Methodology outline

This was the first step of the developed method. Through linear discretization, those vertices in a data file that needed to be interpolated as straight lines or ones that form sharp corners were identified and filtered. To attain such an effect, the distance between each adjacent vertex is determined and normalized (vector magnitude), while maintaining the hierarchical vertex structure of the data file. Those points with a distance greater than a threshold were candidates for linear interpolation. The threshold value was determined by experimentation and sampling on various vertex-based models. If any of the distances were larger than this threshold, the segment corresponding to those vertices was connected by a straight-line and the vertices were stored in a data file. The residual vertices were non-linear candidates, which did not cross the threshold limit, and were considered to be B-Spline interpolated selection points.

limited and not traditionally implemented in collaborative VR systems. These file formats are used in CAD environments and do not translate to a VR environment directly. The proposed method can act as a bridge for these formats as well as those that do not use parametric representations to produce small data files for efficient CV. Figure 1 shows the general outline of the proposed methodology.

Vertex-based models are typically generated from solid modeling packages and to preserve the existing practices, 3D solid model data generated from one of these programs was used in this research. X, Y, and Z vertices of a solid model were extracted and discretized into two components – linear and non-linear. While the linear component handles sharp corners or edges in the vertex-based data, the non-linear component handles curvatures.

Figure 2 identifies potential linear and non-linear components in a solid model representation. The left part of Fig. 2 shows a part of the wire-frame representation of a solid block, focusing on two adjacent vertices that are considerably far apart, forming a potential linear component. The right part of Fig. 2 shows a wire-frame section of the block that has adjacent vertices quite close together and is a possible non-linear candidate. Such non-linear vertices can be interpolated with parametric B-Splines.

A. Linear Discretization

A suitable mechanism to delineate the vertex data as linear and non-linear needed to be chosen.

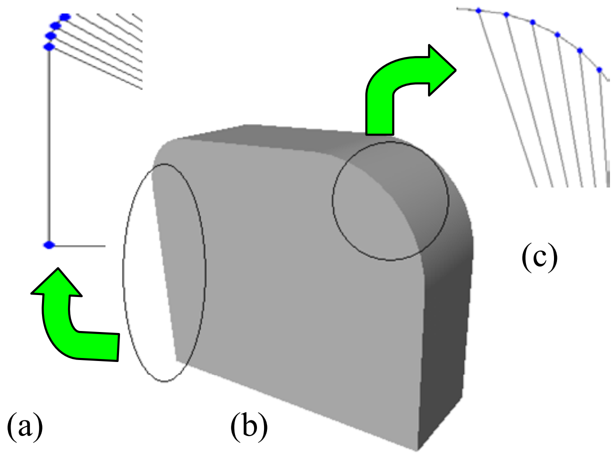

Figure 2 - A vertex-Based block distinguishing Linear and Non-Linear features

B. Non-Linear Discretization

The residual vertices, after linear discretization, were used for non-linear processing. Each of these groups was separated by linear components, if they exist, and were interpolated as separate parametric B-Spline curve segments. The calculation of a polynomial equation that described the behavior of a curve was sufficient to represent its curvature from a complex 3D model. The polynomial equations developed were chosen to be parametric in nature because of their ease in handling and manipulation. More about the characteristics and terminology of B-Splines can be found in Rogers[12] and Lee[13].

A B-Spline is computed by the parametric equation,

$$P(t) = \sum_{i=0}^{n} B_i N_{i,k}(t) \qquad (u_{k-1} \le t \le u_{n+1}) \tag{1}$$

$$N_{i,k}(t) = \frac{(t-u_i)N_{i,k-1}(t)}{u_{i+k-1}-u_i} + \frac{(u_{i+k}-t)N_{i+1,k-1}(t)}{u_{i+k}-u_{i+1}} \tag{2}$$

$$N_{i,1}(t) = \begin{cases} 1 & u_i \le t \le u_{i+1} \\ 0 & otherwise \end{cases} \tag{3}$$

The exact methodology developed uses B-Spline interpolation to generate the curve segments from the original vertices. Control points, which control the overall shape of the B-Spline, were calculated for a given group of non-linear vertices. Then, Cox[14] and de Boor's[15] recursive functions were used to approximate the curve segment. In this manner, the curve exactly passed through all of the original vertices. In addition, non-periodic knot vectors were also used so that the computed curve passed through the first and last vertices to maintain continuity between each subsequent curve segment.

The filtered non-linear points obtained through the linear discretization stage described in the previous paragraphs were sampled into separate groups and B-Spline interpolation techniques were applied to each group. The following matrix equations were used for the interpolation.

$$[P] = [B][N] \tag{4}$$

$$[P][N]^{-1} = [B][N][N]^{-1} \tag{5}$$

$$[P][N]^{-1} = [B] \tag{6}$$

Equation (4) is the matrix form of (1). In B-Spline interpolation, points on the curve are known quantities (i.e., *P(t)* is known). The basis function $N_{i,k}(t)$ can be obtained using a B-Spline recursive relation from Eq. (2) and Eq. (3). The control points, B_i are unknown quantities and the matrix representation of the system (Eq. 4-6) illustrates the procedure to back calculate these control points. Cox and de Boor's relations are then used to obtain a B-Spline from these newly obtained control points which in turn, makes the curve to pass through the initially considered set of vertices for each non-linear group.

The selection of suitable parametric *'t'* values for the Basis Matrix calculation using the B-Spline recursive relations in Eqs. (2) and (3) is crucial for proper curve interpolation. Several methods are available to perform such interpolation but each comes at an expense of accuracy or the inability to handle large-scale datasets. Shene[***] compares the performance of various interpolation methods, and from all the sampled methods, the Universal Method[16] was chosen to be a very convenient approach for B-Spline Interpolation. As opposed to a traditional approach of B-Spline interpolation where parameters are first determined, then followed by the computation of knot vectors, the Universal method prescribes the use of uniformly spaced knots for the computation of the parameters. Let *(n+1)* parameters be required, one for each data point, and the degree of the B-Spline curve chosen be *p*. The number of knots chosen would be *m+1*, where *m = n+p+1*. These knots are uniformly spaced. More precisely, the first *p+1* knots are set to 0, the last *p+1* knots are set to 1, and the remaining *n-p* knots evenly subdivide the domain [0, 1]. Therefore, the knots are given by:

[***] Web Reference: Shene, C., http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/notes.html

$$u_0 = u_1 = ... = u_p = 0$$

$$u_{p+i} = \frac{i}{n-p+1} \quad for \quad i = 1,2,...,n-p \tag{7}$$

$$u_{m-p} = u_{m-p+1} = ... = u_m = 1$$

This set of *clamped* type knots (non-periodic) defines *n+1* B-Spline functions. Then, the parameters are chosen to be the values at which their corresponding basis functions reach a maximum. The Universal method was found to be quite stable and produced accurate interpolation results.

IV. Vertex Processing Using the Universal Method

The following steps define the proposed methodology to transform vertex based model data to a combination of lines and B-Spline curves.

1. *Determination of the order, k:*

Numerically, the order of a curve is one greater than the degree of the curve. Though the order of a B-Spline curve is independent of the number of control points, it needs to have at least *k* points to have a curve of order *k*. Since typical solid modeling systems use 4th order (3rd degree) curves as *de facto* standards, the maximum value of *'k'* allowed in the scope of this research was set to 4, but scalable to a larger value if required.

2. *Knot Vector, u_j and Parameter 't':*

Knot vectors were calculated first using the Universal Method, followed by the generation of *'t'* values for each of the non-linear curve segments by an iterative sampling process. These values were between 0 and 1. All real numbers between 0 and 1 a *'parameter interval'* apart (i.e. 0.005) were extracted. For example, 200 numbers were generated between 0 and 1 each of which differ by the parameter interval. Cox and de Boor Basis Functions were then calculated using the generated parameters, knot vectors and polynomial order as inputs. Then, the parameter values that corresponded to where the Basis Function reached a maximum were stored (in accordance with the Universal Method). These stored values were the final parametric *'t'* values for each non-linear curve segment. By computing 200 Basis Function values, the best parametric *'t'* values were chosen by an iterative sampling process.

3. *Basis Function calculation:*

The previous steps computed Basis Functions only for the purpose of sampling and determining the parametric *'t'* values. However, once these *'t'* values were obtained, the actual Basis Functions were computed, once again, by the use of Cox and de Boor's recursive functions.

4. *Solve for the Control Points:*

LU Decomposition[17] was used to solve matrix Eq. (6) to obtain the values of the control points.

5. *Regeneration of B-Spline:*

The polynomial order, knot vector, parametric *'t'* values and control points that correspond to each non-linear vertex-group were then available. A B-Spline was then generated using these values as inputs in Eqs. (1-3). The resulting B-Spline exactly passes through the original non-linear vertices. The matrix [B] computed from step 4 corresponds to the coefficients of Eq. (1). These coefficients computed for all non-linear components were appended to an output file containing the linear component information.

6. *Intermediate fillers:*

The discretization stages separated the vertices from the vertex-based model into linear and non-linear components, but did not account for the continuity between these groups. Intermediate fillers were then generated to fill the gaps between two non-intersecting curve segments.

This procedure results in a parametric representation for the original vertex data. The B-Spline interpolated curve segments internally have a degree of continuity equal to the degree of the curve. The linear, non-linear and the filler segments maintain C^0 continuity between each other.

V. Hierarchical Vertex Ordering

This is the primary manner in which the vertex data is input into the software implementation of the developed method. The method makes use of the hierarchical vertex structure of the vertex-based data file, which originally was generated from a solid modeling package for this research. In this method, the vertices in a data file are extracted and the developed interpolation techniques are applied.

In the development of this interpolation model unwanted linear segments were sometimes created. It was discovered that these irregularities were due to the use of a single threshold value for linear and nonlinear discretization. Sometimes linear segments were drawn which were not part of the original model file as ordered vertices were not always joined in the original data. As such, this method was modified through the addition of an intermediate threshold value. This value determined whether each linear segment should be drawn.

Although the results from this method were promising (as will be shown in the results section) additional vertex ordering methods were developed to assess their feasibility. These methods take advantage of planar delineation and feature set information in the original vertex-based data file.

VI. Index-face Vertex Ordering

This method extracts the indices of each group of plane or surface information from the vertex-based data file along with the vertex coordinates. The vertices are then arranged in ascending order with duplicate entries eliminated. The developed interpolation schemes are then applied on each of these groups to regenerate the model.

VII. Looped Index Ordering

In many vertex-based models, surfaces are generated by a set of polygons connected to a common point on a face. A pattern was created by the assumption that each surface is created by multiple connected four sided polygons (quads) or triangles. Two to three vertices connected to a common vertex create these polygons. Thus, if the vertices of each polygon are arranged properly, the perimeter of each surface is known and can be interpolated. For example, if the first triangle, of a face, was built by connecting the first three vertices of the data file, its designation would be (0, 1, 2). The second triangle would then be (0, 2, 3) and so on. Each subsequent triangle was generated in this manner in a counter clockwise (CCW) approach, as most visual models are created in CCW manners so the outward normals point correctly for lighting and shading effects. When this was complete, a list of vertex indices was available for a surface of the model. The interpolation schemes developed were then applied.

VIII. Results and Discussion

Results from two test cases run using the methods described in sections 3.4 – 3.6 are presented in this section.

Table 1 - Vertex information of test cases

Vertex-based Model	Number of Vertices
Ellipsoid	358
Truncated Hershey Kiss	478

Table 2 describes the vertex attributes of models that were used as test cases. The vertices were extracted from the vertex-based file and the developed method was applied. The results using the hierarchical vertex ordering input method are presented in Figs. 3 and 4. All of the input methods demonstrated promising characteristics for the conversion of vertex data to parametric representations. However, no single method produced a complete representation without some visual errors. The solution may be in the combination of the three methods developed to fully leverage the strengths of each.

Regardless of the method used, the savings in file size was significant. Table 2 depicts the file size savings attained for hierarchical vertex ordering. The savings obtained from the other two methods was on the same order of magnitude.

Table 2 - File Size Comparison Table

Name of the Model	Original File Size (Bytes)	Output File Size (Bytes)	Savings Achieved
Ellipsoid	44629	16263	63.56%
Truncated Hershey	55245	20694	62.54%

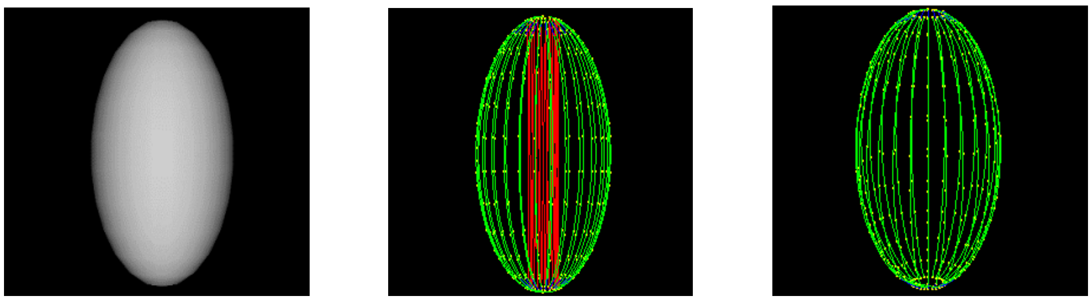

Figure 3. Interpolation results from test case 1
(a) Original vertex-based Solid Ellipsoid Model.
(b) Linear and Non-linear interpolated representation.
(c) Modified Interpolation results with intermediate threshold.

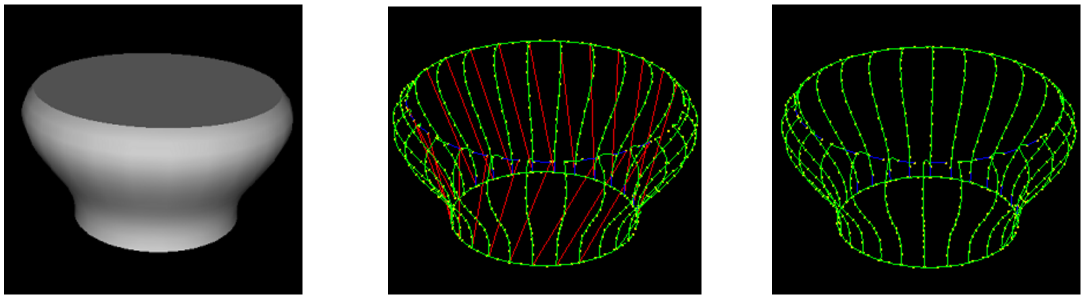

Figure 4 - Interpolation results from test case 2
(a) Original vertex-based Solid Truncated Hershey Kiss Model.
(b) Linear and Non-linear interpolated representation.
(c) Modified Interpolation results with intermediate threshold.

Each test case resulted in more than 50% file size reduction. These results, when viewed for more complex and larger vertex-based datasets, would result in megabytes of savings. Further improvement of this method will only increase these savings. It is anticipated that a detailed model can be decreased to a size of kilobytes to a megabyte, more than suitable for real-time interaction.

IX. Real-Time Interaction

In a collaborative scenario, design changes are made to vertex-based models and it is a common norm to transmit the entire altered model between collaborative designers through the Internet. This results in lower real-time interaction since typical visual design data is very large. However, using the developed parametric interpolated data models, local parameters and coefficients that occupy a couple of Kilobytes can be extracted and transmitted either to a collaborative peer or the server reflecting local alterations to the visual data. This retains the real-time interactivity in a collaborative VR environment. A one time transmission of the full dataset, still extremely small in comparison to the original vertex data, may be necessary.

X. Web-based Content Delivery

It is essential that collaborative clients interact with visual data that looks exactly the same on any computing platform. The models developed in this research are platform independent, and the visual output currently supports two operating systems SGI Irix and Linux, with versions for Windows, Solaris, and MacOS in development. It is also important that the client users have a very small learning curve for operating the visual client. The output data file format is designed in such a way that a lightweight visual client can be built and easily delivered, platform independently, over the Internet through an ordinary web browser. No specialized software is necessary to be installed on a client nor is there are any scripting languages or complex software interfaces to learn. Through a web-based collaborative environment a user is able to access the visual representations in real-time. The main issues that were addressed were real-time access to the data and platform-independency. The B-spline conversion method ensures that the size of each visual representation was kept sufficiently small. Allowing true platform-independence is another matter altogether. Several technologies for this are currently available including Microsoft Corporation's .NET (http://msdn.microsoft.com/webservices/) environment and Sun Microsystems JavaONE (http://java.sun.com/webservices/index.jsp). Broadly, these technologies encompass what has been termed "web services". Web services are the next generation of Internet based data collaboration. Unlike current web sites, which are mostly static displays of data, web services promises full interactive applications delivered over the Internet. The current implementation of web services involves a steep learning curve to create them (the development environments are very complicated), as well as specific hardware and software requirements on both the client and server computers. Even though web services have been touted as "build software once and deploy it anywhere", in actuality there are a number of issues that must be resolved before this is truly possible.

In this research a modified web services paradigm was used. Essentially, by building software several times (for different platforms) and delivering it through truly platform-independent Internet channels, a user can access a full-featured software application from any computing platform and any geographic location. The delivery method chosen was a web-based system that operates on all workstations and computers and most Personal Digital Assistants (PDAs). The environment is named the Interactive Virtual Environment (IVE). This environment leveraged heavily off work performed by Winer and Bloebaum[18]. IVE runs on most web-browsers including Netscape, Internet Explorer, Mozilla, Safari, Konquerer, and Opera. It was designed to function over a range of Internet connection types from broadband to dial-up always with real-time or near real-time interaction. Figure 5 shows a screenshot of IVE.

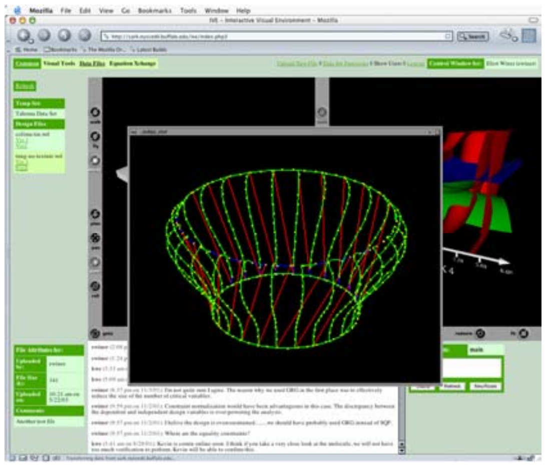

Figure 5 – Collaborative System showing common meeting area

A scientist logging in is presented with various layers in which to work. One layer is the common area (shown in Figure 5). This is where all users can communicate through text-based chat, exchange/view data and share simulation results. A user may also go to a "Visual Tools" layer where the lightweight visual client (QuickView) is available (also shown in Figure 5). This client is what displays the data produced by the proposed method in this paper. Here, the user's machine is automatically queried and a QuickView installation is customized for that particular workstation. The software is downloaded to the user's machine, installed (after being granted permission), and launched for use. In this manner, many issues with traditional software installation are eliminated. A third layer provides a workspace to examine files and make notes and observations. This is somewhat offline from discussions and interactions that may be taking place elsewhere. Additional layers for project management and organization are currently being developed. At any time in IVE, a user may view a calendar of events, see who else is logged in and perform detailed data management.

These features allow for data and communication to occur in a variety of channels to boost ideas and improve the quality of decisions made for complex engineering design. The system has been tested several times with a large number of users (approximately 340) separated by large geographic distances (Columbia, South America to Iowa, United States). In all tests, users all had real-time interaction regardless of Internet connection, computational hardware, or software.

XI. Conclusions and Future Development

In this paper, a new data model and handling strategy is presented that allows real-time collaborative visualization between geographically distinct parties in the context of a typical VR system. This model includes a novel method for taking a vertex based wire frame visual model and transforming it to a parametric based representation that can be stored at a fraction of the original file size. This representation can be transported over the Internet and viewed with a lightweight visual client on a remote user's desktop through a novel web-based collaborative engineering design system.

While the research presented is promising, there is significant future development which still must occur. The developed method must be extended so that solid models are created from the vertex data with lighting and shading effects included. Also, the methods developed could be built to increase accuracies and avoid redundancies. An appropriate binary file format containing the linear and non-linear interpolation data needs to be developed that would be shipped to the client machine from the server in a collaborative scenario. This will result in additional file size savings.

References

[1]Leigh, J., & Bailey, S., "A Methodology for Supporting Collaborative Exploratory Analysis of Massive Data Sets in Tele-Immersive Environments", Eighth IEEE International Symposium on High Performance Distributed Computing, Redundo Beach, California, Aug 3 – 6, 1999.

[2]Suleiman, M., Cart, M., & Ferrie, J., "Serialization of Concurrent Operations in a Distributed Collaborative Environment", Proceedings of the International ACM SIGGROUP conference on supporting group work: the integration challenge, November 1997.

[3]Winer, E.H., Bloebaum, C.L., "Using the World Wide Web to Employ Concurrent Design Methodologies", Proceedings of 3rd World Congress of Structural and Multidisciplinary Optimization (WCSMO-3), Amherst, NY, May 17-21, 1999.

[4]Isenhour, P., Begole, J., Heagy, W., & Shaffer, C., "Sieve: A Java-Based Collaborative Visualization Environment", IEEE Visualization '97 Late Breaking Hot Topics Proceedings, Oct 22 – 24, 1997, pp 13 – 16.

[5]Boyles, M., Raje, R., & Fang, S., "CEV – Collaborative Environment for Visualization Using Java RMI", in ACM 1998 Workshop on Java for High Performance Network Computing. ACM Press, 1998.

[6]Bajaj, C., & Cutchin, S., "Web based Collaborative Visualization of Distributed and Parallel Simulation", Proceedings of IEEE Parallel Visualization and Graphics Symposium. October 24 – 29, 1999 San Francisco, CA, pp 47 – 54.

[7]Abbott, M., & Jain, L., "DOVE: Distributed Objects Based Scientific Visualization Environment. In ACM 1998 Workshop on Java for High Performance Network Computing. ACM Press 1998.

[8]Garland, M., Heckbert, P., "Surface Simplification Using Quadric Error Metrics", SIGGRAPH, '97, Aug 3 – 8, 1997.

[9]Luebke, D., Erikson, C., "View-Dependent Simplification of Arbitrary Polygonal Environments", SIGGRAPH, '97, Aug 3 – 8, 1997.

[10]Hoppe, H., "Surface Reconstruction from Unorganized Points", PhD Thesis, Department of Computer Science and Engineering, University of Washington, June 1994.

[11]Chu, T., Fowler, J. E., & Moorhead, R. J. II., "Evaluation and Extension of SGI Vizserver", in "Visualization of Temporal and Spatial Data for Civilian and Defense Applications III, G. O. Allgood and N. L. Faust, Eds., Orlando, FL, April 2001, Proc. SPIE 4368, pp 63 – 73.

[12]Rogers, D., "An Introduction to NURBS with Historical Perspective", Morgan Kauffman Publishers, 1999.

[13]Lee, K., "Principles of CAD/CAM/CAE Systems", Addison Wesley Longman, Inc, 1999.

[14]Cox, M. G., "The Numerical Evaluation of B-Splines", J. Inst. Maths. Applics., Vol 15, pp. 95 – 108, 1972.

[15]De Boor, C., “On calculating with B-Spline”, J. of Approx. Theory, Vol. 6, pp. 52 – 60, 1972.

[16]Lim, C., “A Universal Parameterization in B-Spline Curve and Surface Interpolation”, Computer Aided Geometric Design, v. 16 n.5, pp. 407 – 422, June 1999.

[17]Press, W., Teukolsky, S., Vetterling, W., & Flannery, B., “Numerical Recipes in C”, Cambridge University Press.

[18]Winer, E.H., Bloebaum, C.L., “Using the World Wide Web to Employ Concurrent Design Methodologies”, Proceedings of 3rd World Congress of Structural and Multidisciplinary Optimization (WCSMO-3), Amherst, NY, May 17-21, 1999