

A Method for Sharing Interactive Deformations in Collaborative 3D Modeling

Hiroaki NISHINO*

Atsunori SAKAMOTO

Dept. of Computer Science and Intelligent Systems,
Oita University, Oita 870-1192, JAPAN

Tel : *+81-97-554-7876

E-mail : *hn@csis.oita-u.ac.jp

Kouichi UTSUMIYA

Kazuyuki YOSHIDA

Kazuyoshi KORIDA

Dept. of Communication,
Oita Pref. College of Arts and Culture
Oita 870-0833, JAPAN

ABSTRACT

This paper proposes a new approach to collaboratively designing original products and crafted objects in a distributed virtual environment. Special attention is paid to concept formulation and image substantiation in the early design stage. A data management strategy and its implementation method are shown to effectively share and visualize a series of shape-forming and modeling operations performed by experts on a network. A 3D object representation technique is devised to manage frequently updated geometrical information by exchanging only a small amount of data among participating systems. Additionally, we contrive a method for offloading some expensive functions usually performed on a server such as multi-resolution data management and adaptive data transmission control. Client systems are delegated to execute these functions and achieve "interactivity vs. image quality" tradeoffs based on available resources and operations in a flexible and parallel fashion.

Keywords : 3D object modeling, collaborative design, distributed virtual environment, computer graphics.

1. INTRODUCTION

A distributed virtual environment (DVE) is a technology that has the potential to change traditional design and production methodologies. The traditional methods divide a production process into subprocesses in which only skilled persons are involved. The introduction of DVE technology, however, enables various people such as designers and developers in different fields, external consultants and even customers to be involved in collaborative design and production activities. Even if they are geographically separated, DVE provides them with opportunities to be aware of others' presence and interact in a virtually shared space without any temporal or spatial restrictions. There have been some high-level frameworks and application systems proposed to make DVE a usable and practical technology such as FIVE [17], NPSNET [10], DIVE [6], MASSIVE-2 [5] and Shastra [1]. There

are, however, few trials in which to apply DVE technology, especially for the early conceptual design of new products such as the creation of original object shapes and patterns.

This paper proposes a new approach to constructing a multi-user collaborative environment, especially for conceptualizing and designing new products and crafted objects. Attention is paid to Japanese ceramics, a typical handicraft, as a target application for multi-user collaboration. In traditional art and craft fields, professional craftsmen tend to follow their conventional ways for hundreds of years and to protect their design styles from others. It should, however, be quite useful to share not only digitally-archived art works, but also cultivated skills and experiences. Therefore, a framework for sharing such undocumented knowledge is developed. It provides a set of services to visualize a series of experts' modeling operations, including which primitive shapes to use, how to combine and blend them, and how to deform them to make a new object rather than just sharing a final form. Because many shapes and patterns are created or modified in the course of the modeling process, these intermediate shapes need to be shared efficiently.

On the other hand, off-the-shelf 3D data exchange protocols like VRML are inappropriate for sharing 3D geometric objects whose shapes are rapidly deformed. They are ideal for sharing static objects such as trees and buildings in a virtual space as well as objects like 3D avatars whose positions and orientations are updated while their shapes do not drastically change. Therefore, we extended our intuitive 3D modeling environment [12][13] so as to be usable on a network, enabling multiple participants to work on the same objects and share all intermediate information during real-time collaboration. As shown in figure 1, the system allows the designers to specify shapes and deformation patterns by using their bimanual gestures captured with a pair of instrumented gloves (Virtual Technologies' CyberGlove). The designers can concentrate on the design tasks by using intuitive hand gestures directly translatable into modeling operations. The designated shapes and deformed objects are projected onto a 200-inch large arch screen stereoscopically to give them a sense of presence. To tolerate the variance of human gesticulation, neural networks are utilized to learn and recognize the required gestures for the modeling operations [14][15].

In the real world, professional craftsmen design and produce their art work in a serial routine. They can, however, acquire new modeling strategies through the sharing of other participants' feedback and disparate design skills. Moreover, all participants in a group can share

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

VRST 99 London UK

Copyright ACM 1999 1-58113-141-0/99/12...\$5.00

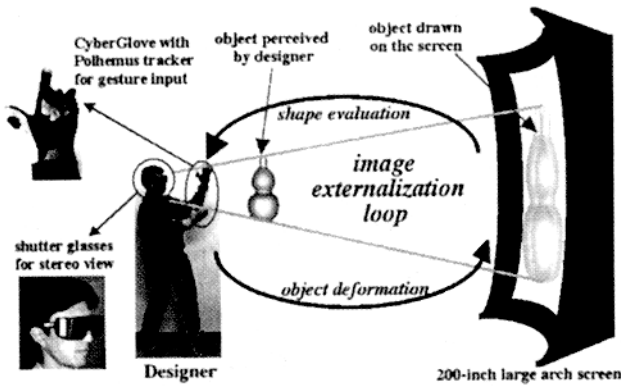


Figure 1 : Gesture-based 3D object modeling system.

the design know-how of the craftsmen through the visualization of their modeling operations, which are ordinarily performed in isolation. They can also participate in design activities through wireless communication links even if they have no wired links to the Internet. Real-world assets like historic relics and exhibited objects in museums can be digitally archived by a portable 3D digitizer. Thus, these things can be shared and utilized in a collaborative design space. The outcomes of the design work can be sent to the rapid prototyping system via the network to manufacture real models. A long-term goal of our research is to organize a synthetic virtual environment, a so-called *seamless real-virtual production factory (SRVPF)*, which supports whole design and manufacturing activities on the network.

2. CHARACTERISTICS OF THE PROPOSED METHOD

An essential problem with dynamic 3D object sharing in the DVE is to realize adequate “image quality vs. interactive response” tradeoffs without sacrificing scalability. As shown in figure 2, a typical approach adopted in many existing systems tends to depend on server’s data management capabilities and network bandwidth to achieve the goal. A majority of the 3D geometric data format shared and transmitted on the network is polygon mesh. Because it tends to be large in size, it is critical to reduce it for data transmission. There are some proposals to change the resolution and size of the shared data according to the network bandwidth and client’s performance [9][11]. Multiple data in different resolutions are generated beforehand, and coarser data are transmitted for the narrow-band links and the slow clients. This approach, therefore, causes a serious loss of quality for the client who has a low-speed link or CPU. A participant’s sense of presence is significantly affected by the low-quality image. The adaptive resolution control also increases a server’s responsibility to monitor and administer the network load and the client performance on the fly. While the LOD (level-of-detail) [7] and the progressive meshes [8] are available for multi-resolution data management, they are too costly to compute multiple resolutions for rapidly changing 3D objects.

Figure 3 shows our proposed data sharing method. The idea is to offload a server’s burden by using clients more beneficially. Because personal computers and graphics hardware have dramatically improved cost performance ratio, they have the capability to compute expensive 3D geometric and rendering algorithms for interactive applications. Therefore, we employed implicit surface representation to share and visualize 3D objects. Because our proposed method can

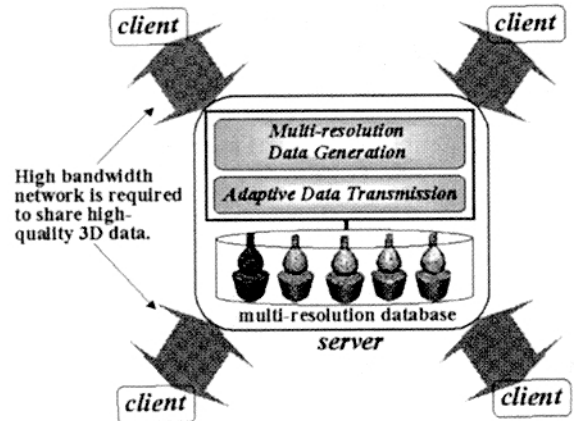


Figure 2: Server-dependent data sharing method.

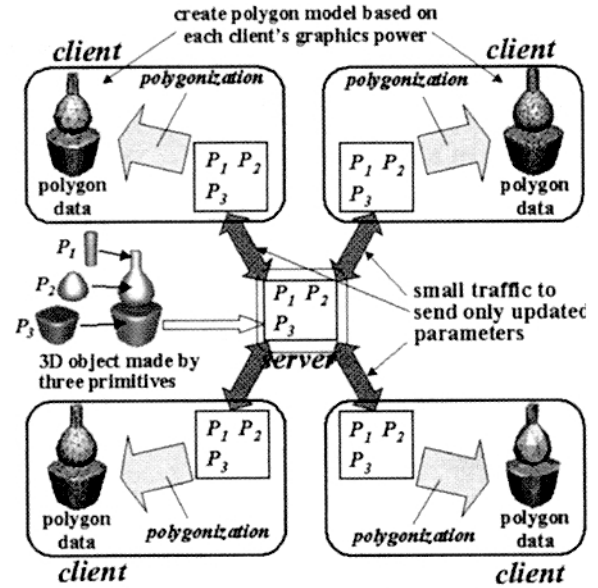


Figure 3: Proposed data sharing method.

express various shapes by using a few dozen functional parameters, only these parameters need to be shared among the clients on the network. If a modeling operation is performed on a client, only modified parameter values need to be transmitted, not the whole set of parameters. It is the client’s responsibility to convert the set of parameter values to a ready-to-visualize data format such as the polygon mesh. Each client generates a 3D model in an appropriate resolution according to its CPU and graphics power. It effectively utilizes clients’ computing power in parallel, making the system scalable.

3. DYNAMIC 3D OBJECT MODELER

3.1 Basic Modeling Functions

A modeler implemented in the system represents 3D primitive shapes by introducing the superquadric function [2] whose distinctive feature is its small number of parameters for expressing various shapes as shown in figure 4(a). The function is extended to make natural deforming operations possible, such as those illustrated in figure 4(a). Further complex shapes are computed and rendered as implicit surfaces by blending multiple primitive shapes. As depicted in figure 4(b), the blending yields smooth and interesting shapes that are diffi-

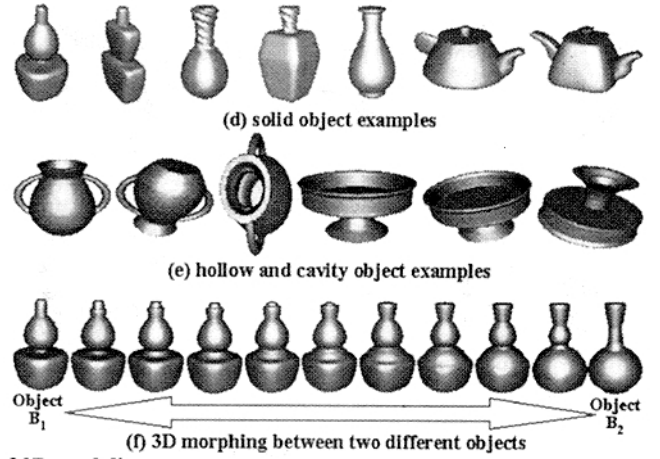
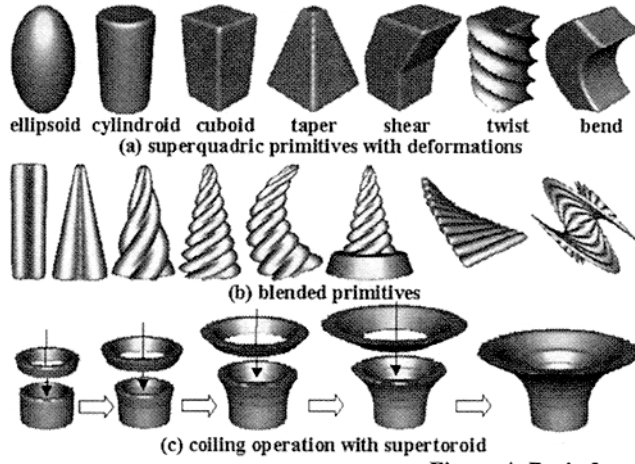


Table 1: Primitive parameter values of the bottle in figure 3.

	type	r_x	r_y	r_z	r_a	e_1	e_2	d_{TPX}	d_{TPY}	d_{SH}	d_{TW}	d_{BE}	x_0	y_0	z_0	yaw	pitch	row	rfu
P_1	0	0.3	0.3	1.0	0.	0.1	1.0	0.	0.	0.	0.	0.	0.	0.	2.2	0.	0.	0.	1.0
P_2	0	1.0	1.0	1.35	0.	1.0	1.0	-0.35	-0.35	0.	0.	0.	0.	0.	0.75	0.	0.	0.	1.0
P_3	0	1.25	1.25	1.4	0.	0.15	0.65	0.15	0.15	0.	0.	0.	0.	0.	-2.12	0.	0.	0.	1.0

type: primitive type (0: ellipsoid, 1: torus)

d_{xx} : local deformation strength (TPX, TPY: tapering; SH: shearing; TW: twisting; BE: bending)

yaw, pitch, row: rotation parameters

x_0, y_0, z_0 : local center coordinate

rfu: reserved for future use

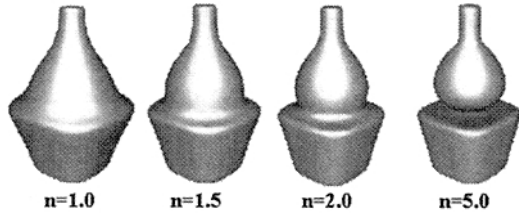


Figure 5: Blending operation with different field strength n .

Table 2: Blending parameter values of the bottle.

D_{TPX}	D_{TPY}	D_{SH}	D_{TW}	D_{BD}	FS
0.	0.	0.	0.	0.	3.0

D_{yy} : global deformation strength

(TPX, TPY: tapering;

SH: shearing; TW: twisting;

BD: bending)

FS: field strength for global blend

3.2 Mathematical Expressions

The mathematical expressions of the deformable-superellipsoid P_e and -supertoroid P_t are as follows:

$$P_e(x, y, z) = \left[\left\{ \left(\frac{x}{r_x} \right)^{2/e_1} + \left(\frac{y}{r_y} \right)^{2/e_1} \right\}^{e_1/e_1} + \left(\frac{z}{r_z} \right)^{2/e_1} \right]^{e_1}, \quad (1)$$

$$P_t(x, y, z) = \left[\left\{ \left(\frac{x}{r_x} \right)^{2/e_1} + \left(\frac{y}{r_y} \right)^{2/e_1} \right\}^{e_1/2} - \frac{r_a}{\sqrt{r_x^2 + r_y^2}} \right]^{2/e_1} + \left(\frac{z}{r_z} \right)^{2/e_1} \right]^{e_1}. \quad (2)$$

where r_x, r_y , and r_z are the scale parameters to control the size of the primitives; r_a is the torus radius; e_1, e_2 are the squareness parameters to control the shape. The shearing among the deforming operations as shown in figure 4(a) can be realized by the following function to transform a vertex (x, y, z) in equation 1 and 2 into a new vertex (x', y', z') .

$$x' = \frac{k_{SH} z}{r_z} + x \quad (z \geq 0) \quad \text{or} \quad x' = x \quad (z < 0), \quad y' = y, \quad z' = z \quad (3)$$

where k_{SH} is the parameter to amplify the shearing operation. The equations of other deformations can be found in [12]. Each primitive has nineteen parameters in total such as those described in table 1. It defines a bottle shape made of three primitives as shown in figure 3. Then, a complex object blended by m primitives is defined as follows:

cult to make by using polygon meshes. The deforming operations are applicable to the blended objects as well.

Our previous modeler was only able to represent solid objects because it adopted the superellipsoid [2] as a sole primitive function. The supertoroid primitive [2] is added to enable the modeler to express hollow and cavity shapes, and the coiling interface is implemented to easily create these shapes as shown in figure 4(c). Whereas figure 4(d) shows some solid shape examples, figure 4(e) illustrates a vase and a plate made of supertoroid primitives. Because 3D object creation is a difficult task, we also implement a function to generate new shapes from existing ones by applying a 3D morphing technique. Figure 4(f) shows a series of 3D shapes generated in the morphing between the two different objects.

Because of these enhancements, the modeler has the superior ability to easily express natural and smooth shapes such as crafted objects. In addition, it can express various kinds of art works and human bodies. Consequently, it allows the designers to easily describe their images and conceptions by replacing them with primitive shapes and deforming operations, presenting them as virtual 3D mock-ups rather than 2D sketches. The modeler cannot, however, be used for the precise recovery of complex objects and sophisticated product design. Supporting these tasks requires the integration of other data representation methods like parametric surfaces.

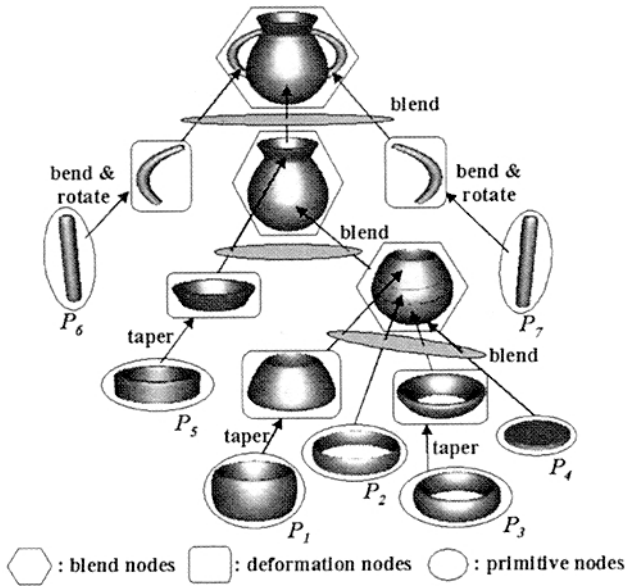


Figure 6: Tree-structured representation of 3D object.

$$B(P_1, \dots, P_n) = 1.0 - (P_1^n + P_2^n + \dots + P_n^n)^{\frac{1}{n}}. \quad (4)$$

where n is used to control the smoothness of the blending operation. Figure 5 indicates the effect of this parameter. The blending operation is defined by six parameters as described in table 2. The morphing between two blended objects B_1 and B_2 is defined as follows:

$$B_{morph} = t \cdot B_1 + (1 - t) \cdot B_2, \quad 0 \leq t \leq 1. \quad (5)$$

4. EXTENSION INTO SHARED ENVIRONMENT

4.1 Shared Object Data Structure

To make the 3D modeler operable in a distributed collaborative environment, a mechanism for efficiently sharing, maintaining and updating a set of parameter expressions among the participating systems is required. Additionally, functions to display some parts or previous shapes of an object are necessary to allow the participants to focus their attention on a specific portion of the object and review the modeling process. As shown in figure 6, a tree-structured object representation method is developed and implemented to satisfy these requirements. The figure illustrates a modeling process of the vase as shown in figure 4(e). The leaves of the tree correspond to the primitive shapes used to form the vase. Each leaf holds a set of primitive parameters as shown in table 1. The local deformations applied on each primitive such as tapering, bending and rotation are recorded in a unary node. It holds the up-to-date parameter values corresponding to these operations. A blend of n primitives is defined as an n -ary node. It keeps a set of blending parameters as mentioned in table 2.

Figure 7 shows the internal data structure of the tree. It begins with a link to all primitives and grows toward the root of the tree. The whole modeling process can be visualized by traversing this data structure and converting a functional expression into a polygon mesh. Each primitive may be made invisible or be returned to its previous shape by turning the conversion off for that node. To realize a high-speed conversion engine, the implicit polygonization algorithm proposed

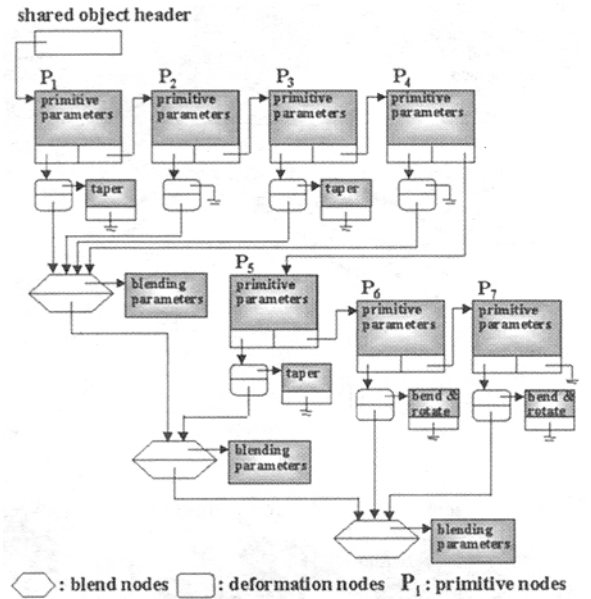


Figure 7: Internal data structure of the tree in figure 6.

by Bloomenthal [3] was chosen among several methods because of its performance. It can produce a polygonal surface to approximate the implicitly defined object faster than the other methods. We made some extensions for adaptive visualization which are described later.

4.2 3D Object Sharing Method

To constitute a collaborative modeling environment on the network, a system shown in figure 8 is designed and implemented. All clients have their own replicas of the 3D modeler and tree structure to do the conversion in parallel. Additionally, the system can easily support a heterogeneous distributed environment, and it can customize the functions based on the available resources for each client.

Because all clients have copies, how to resolve simultaneous updates on shared 3D data is a critical issue. A system like Shastra allows multiple participants to simultaneously modify different parts of an object [1]. It, however, demands a complex mutual exclusion mechanism which easily degrades the performance in heavily loaded environments. In addition, simultaneous modification seems to be useful at the detail design stage because overall specifications and structural relationships between the parts of the object are clearly defined. It might be easy to divide a design task into subtasks done in parallel. It would not, however, be suitable for the early design stage because the entire structure is not fixed nor ready for subdivision. A more relaxed sharing strategy to allow a single performer at a time seems natural. Therefore, shared object modification is controlled by an update right (UR) managed by the server. The participants compete for the UR before doing their desired deformations. A series of deforming operations performed by the UR holder is called "modeling unit." The UR is effective until the holder releases it.

Figure 8 shows a data sharing procedure implemented on the basis of the above design policy.

- (1) First, all participants login to get connections to the shared environment as labeled **a** in figure 8 (**8-a** for short). The latest object data are returned with an ACK response from the server to the requesting client (**8-b**).

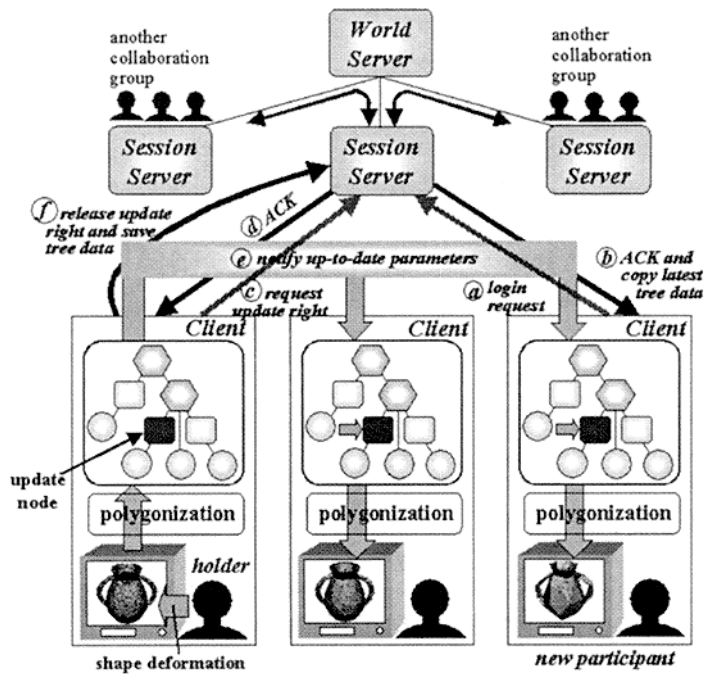


Figure 8 : 3D data sharing mechanism and procedure.

- (2) Next, participants wanting to execute modeling tasks compete for the UR of a target object through the server (8-c). If only one client is requesting the UR, then the server immediately returns an ACK with the latest object data and login information (8-d). Otherwise, one client is selected based on predefined priorities or time stamp. If there are several objects in the area, each object can be updated simultaneously.
- (3) Then, the winning client has the sole right to perform any modeling operations. All of his/her operations are directly broadcast to all clients. The data packet only includes the updated parameters to keep the network traffic low. Each client initiates the polygonization to visualize the modified shape in an appropriate resolution (8-e).
- (4) Finally, the UR holder notifies the server of the release with the updated data (8-f).

As concerns the data transmission between the server and clients, it depends on the contexts whether reliability is more important than speed. For example, the data exchanged in the above procedures (1), (2) and (4) are handled by the TCP protocol because they need reliability. On the other hand, the update information broadcast in procedure (3) is treated differently according to the characteristics and the requirements of the collaboration environment as follows:

- use multicast for many unspecified participants joining from remote locations,
- select UDP for the collaboration on a LAN, or
- choose TCP for reliable data transfer.

The latest data are transferred from an old to a new UR holder through the server by TCP to avoid any data loss. Accordingly, the server always holds up-to-date information on the latest modeling unit, backing it up for failure recovery and new participants' arrival. Scalability is another critical issue to realize a successful DVE system on the Internet. Hence, hierarchical client-server architecture is employed

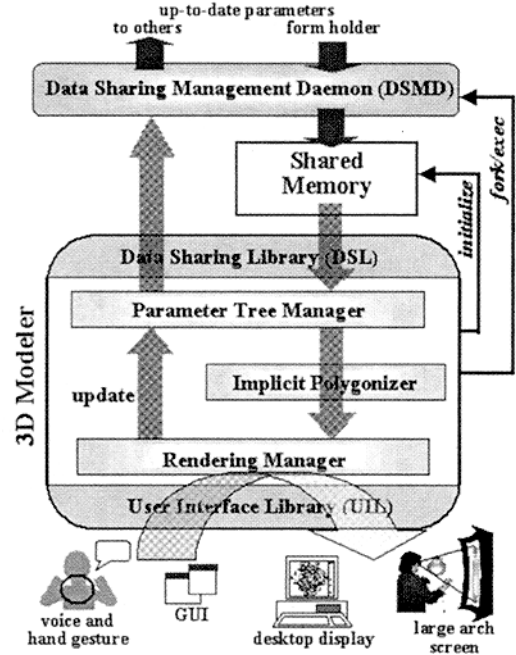


Figure 9 : Client system structure.

to support a large-scale virtual environment as shown in figure 8. Each participant selects a session where the design of his/her target object is in progress. While the session is similar to the "cell" in NPSNET [10] and the "group" in MASSIVE-2 [5] which have a corresponding physical structure like a multicast group, it additionally offers a way to define a party aiming at a specific design task. The server grants the UR to its directly administered participants to make the UR management a simple yet efficient procedure. The participants residing in other sessions are only allowed to look at the latest data. They can move the session by logging out from the current one and reconnecting to the new one. The world server is responsible for the whole system configuration.

4.3 Update Notification and Visualization

The modeling operations performed by a UR holder need to be shared as precisely as possible to faithfully reproduce and efficiently exploit participants' skills to create 3D shapes. To satisfy this requirement, update information is sent whenever a modeler's atomic operation, such as a single deforming action or a primitive addition or deletion, is executed. While a primitive addition is notified by broadcasting nineteen parameters, a primitive deformation can be announced by at most three parameters (i.e. three for size, location or orientation change, and one for deformation).

Figure 9 shows the client system structure. The rendering manager reiterates a drawing function to output the latest data. It checks whether any update information has arrived in every rendering cycle and renews the tree with the received parameters if necessary. Then, it initiates the implicit polygonizer to compute and visualize the updated geometry. Figure 10 illustrates the polygonization method. It enfolds the object with uniform-sized cells intersecting with the object's surface, computing the functions on every cell vertices. Then, it checks cell edge polarity to detect the transverse edges. The transverse edge is an edge that has corners of differing polarity, like the edge $v_i y_k$ in the figure. Because output resolution and computation time can be

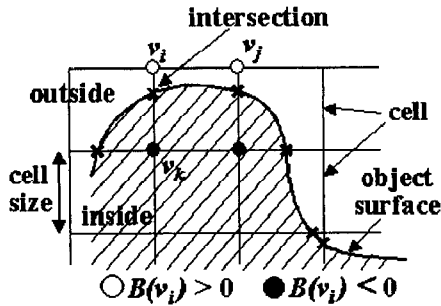


Figure 10: Cell polygonization mechanism.

adjusted by changing cell size, the following enhancements are incorporated into the modeler to support adaptive polygonization according to each client's performance and modeling operations:

- One function is to adjust the cell size to complete the polygonization within the specified time. It guarantees fixed response time to get the result during interactive deformations.
- Another is to compute the result at the predefined number of polygons. This function is useful for shape evaluation by generating high-quality images.

The modeler always is watching the polygonization time and the number of generated polygons to make the above adjustments. When these measures exceed the thresholds, the modeler changes the cell size until they return to the allowable limit. Because a polygonization is only activated when the tree is updated, client's local operations, such as viewpoint movements, are quickly handled. The parameters are delivered to the modeler via a shared memory region to prevent the rendering from being disturbed by the data transmission.

5. PROTOTYPE SYSTEM AND INITIAL EXPERIMENTS

A prototype system is implemented in the hardware environment as shown in figure 11. All software components are written in C, C++ and OpenGL. A single server environment is constructed to evaluate the proposed method. Common services such as gesture and voice input handling, GUI interface for desktop systems, stereoscopic projection, and asynchronous data transmission are implemented as a library to be shared among VR applications [16].

When the actual collaborative session is organized, deciding the set of optimal response time and evaluation quality as explained in 4.3 are the most critical tasks. Two model shapes, a vase and a bottle, are used to discuss how these values are determined. The graphs in figure 12 show the relationships between polygonization time and the number of generated polygons measured on three different machines (Onyx2, O2 and Indy). Both graphs indicate the linearity. Because more than 90% of the polygonization time is spent on computing the functions, the time varies in proportion to the number of cell vertices. Hence, it is easy to find the time needed to generate a specific number of polygons and vice versa. In the case of the vase, the polygonization time for interactive deformations is determined as follows:

- (1) Take a few sample data on each machine to plot the graph as shown in figure 12(a).
- (2) Decide the lowest number of polygons to preserve output image

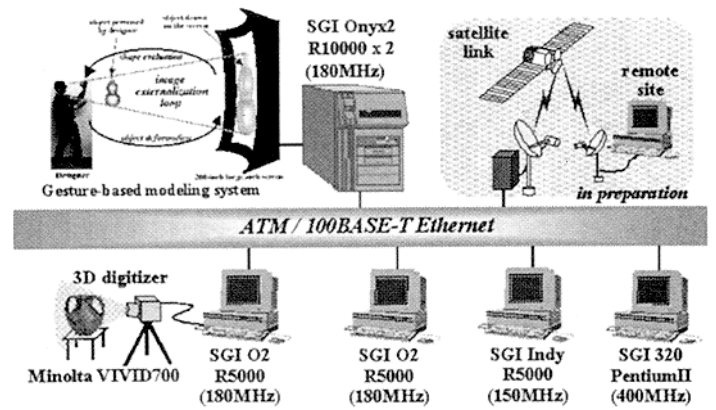


Figure 11 : Prototype system organization.

quality usable for the modeling operations. Here, 2,500 satisfies this criterion.

- (3) Obtain the polygonization time to generate this number on the slowest machine (Indy). This is found from the graph to be 1.8 sec, and 2.0 sec is chosen as the final value by adding some variations.

According to figure 12(a), Onyx2, O2 and Indy can generate 8000, 3400 and 2800 polygons within the 2.0 sec processing time, respectively. Because the delay on the update broadcast can be ignored in our high-speed LAN environment, the output always appears on all machines 2.0 sec after any deformation. This fixed response time can be preserved in a geographically separated environment by using communication channels with fixed delays such as the satellite link. It is achieved by subtracting the delay time from receiver's polygonization time beforehand. The evaluation quality also is determined by considering the time vs. quality tradeoff. For example, highest quality (i.e. 30,000 polygons) may be selected for all machines without consulting performance. Or, coarser quality (e.g. 15,000 polygons) may be chosen for medium to low-end machines. The time and quality for the bottle are decided in the same fashion as shown in figure 12(b). Since the polygonization time is proportional to the amount of functional computation, it varies with the number of primitive shapes and complexity of the function to represent the shapes. Therefore, automating the above procedure to identify the appropriate values for various objects is one possible future enhancement.

The real models as shown in figure 13(b) are made of the corresponding virtual objects in 13(a) for the purpose of assisting objective value judgement of the created shapes. A 3D solid laser plotter, the SLP-3000 manufactured by Denken Corporation, was used to produce these models. Although they took ten to twenty hours, it is beneficial to have these tangible entities for collaborative modeling.

A problem with our proposed method is the substantial deterioration of image quality caused by minimizing the polygonization time. This is because fixed cell size is used to speed up processing, and subtle shapes tend to be lost by enlarging the cell size. There are, however, some solutions to sustain quality such as changing the cell size according to the surface curvature of the polygonized object [18]. The application of such methods needs to be considered by paying attention to performance degradation. Additionally, the influence of a few seconds delay and the serialized deformations enforced on the col-

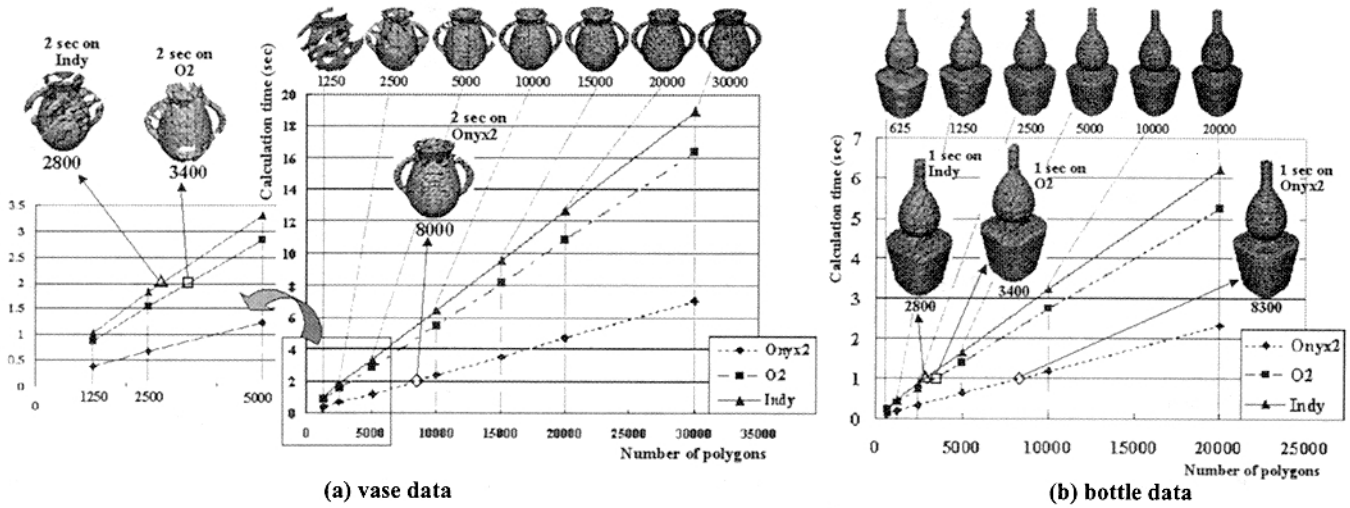


Figure 12: Polygonization time vs. rendering quality.

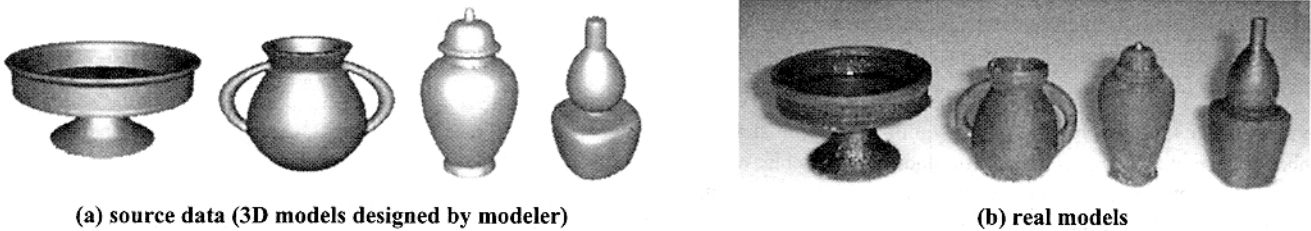


Figure 13: Real models made by solid laser plotter.

laborative modeling needs to be examined from a cognitive science viewpoint.

6. COMPARISONS WITH TRADITIONAL METHOD

This section compares the proposed method with traditional server-dependent data sharing to verify the effectiveness of our method. As illustrated in figure 14, the traditional approach notifies the server of an update event. Then, the server updates the shared data via the object manager and broadcasts the updated data to all clients. Furthermore, the server generates some low-resolution data to deliver to the narrow-band communication links and the low-speed systems. While the server carries out all heavy operations like geometric computation and rendering, the clients receive the ready-to-visualize polygon data. This approach assumes that centralized processing by the server is an advantageous strategy over the distributed execution by the clients. This, however, becomes an abnormal situation due to rapid performance improvement in personal computers. The proposed method allows the clients to manage shared data and generate output with optimal resolutions in parallel without burdening the network and the server as depicted in figure 15. Centralized processing required in the traditional method is replaced by clients' parallel computation.

To estimate the overhead of server's centralized processing, an experiment to measure the sustained performance of multi-resolution data generation was conducted by using the vase data. Figure 16 shows the results measured on Onyx2. A high-performance tool for 3D geometry compression, the so-called *qslim* [4], was used to make the three resolution levels determined for the interactive deformations in the previous section (8000, 3400, and 2800 polygons). The horizon-

tal axis indicates the source data resolutions used for the experiment and the vertical axis shows the time to compress them. While the results vary with the source and the output resolutions, compression of a single resolution takes more than a second. Because this overhead linearly increases in proportion to the number of compressed data files needed to be produced by the server, it becomes a bottleneck. The proposed method, however, allows each client to produce an optimal resolution according to its available resources. Thus it can flexibly cope with the variance.

Table 3 lists the packet size transmitted on the network for the vase and the bottle. It compares the total quantities of parameters to represent these shapes in the proposed method with the ones of corresponding VRML files commonly used for sharing 3D data in the traditional method. The latter ones are made from the results acquired by the 2 sec (vase) and 1 sec (bottle) interactive deformations on three machines as described in the previous section (e.g. a vase shape with 8000 polygons on Onyx2). The geometrical and topological results are converted into the VRML polygonal expression. The proposed method compresses exchanged data into that smaller than 1 percent of the VRML files. While the transmission of the parameters causes no perceivable delays, the VRML file transfer causes 350 msec to one sec delays in the experiment. In addition to server's overhead mentioned above, these delays seriously degrade system performance. Because the update notification is usually achieved by transmitting nineteen parameters (at most) of a target primitive in our proposed method, the effect on the data compression is much bigger than the ratio denoted in table 3.

7. CONCLUDING REMARKS

A new approach to constructing a collaborative design environment

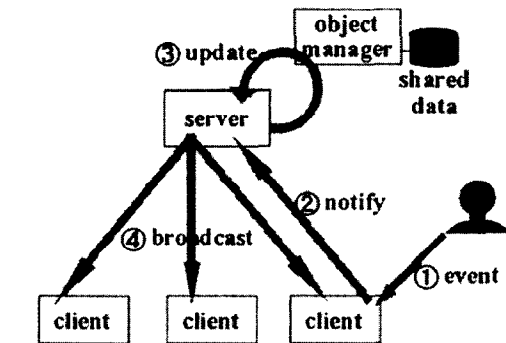


Figure 14: Server-dependent management strategy.

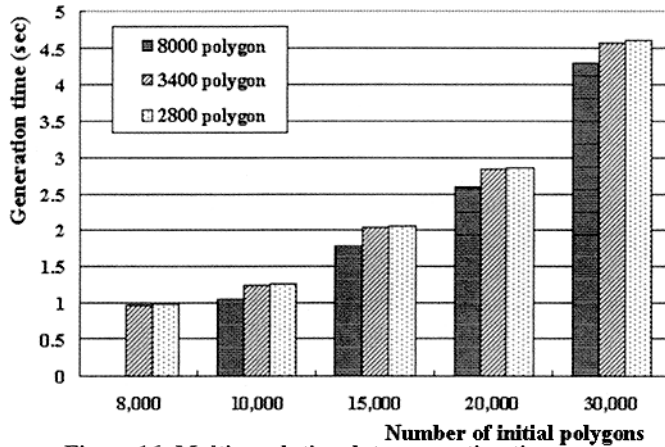


Figure 16: Multi-resolution data generation time.

has been described, and a method for efficiently sharing the process of making concrete 3D shapes from designer's vague images on the network was developed. Rapidly changing 3D shapes are efficiently shared and smoothly visualized by exchanging only a small amount of data among the clients. Additionally, a data sharing strategy based on the tree-structured 3D object representation significantly offloads a server's burden and allows the clients to produce the appropriate output resolutions in proportion to their machine performance. The effectiveness of these ideas was verified through our initial experiments performed on the prototype system.

Conducting further experiments on a wide area network, including wireless links, is imperative. We are currently preparing a satellite link to incorporate into the system. The development of functions to digitally archive and share real-world assets in collaborative design work is another important challenge.

8. REFERENCES

- [1] Anupam,V. and Bajaj,C.L. : Shashtra: Multimedia collaborative design environment, IEEE Multimedia, 1(2):39-49, 1994.
- [2] Barr,A.H. : Superquadrics and angle-preserving transformations, IEEE Computer Graphics and Applications, 1(1):11-23, 1981.
- [3] Bloomenthal,J. : An implicit surface Polygonizer, Graphics Gems IV, AP Professional, pp.324-349, 1994.
- [4] Garland, M. et al. : Surface Simplification Using Quadric Error Metrics, Proc. ACM SIGGRAPH'97, pp.209-216, 1997.
- [5] Greenhalgh,C. and Benford,S.: Supporting rich and dynamic communication in large-scale collaborative virtual environment, Presence, 8(1):14-35, 1999.

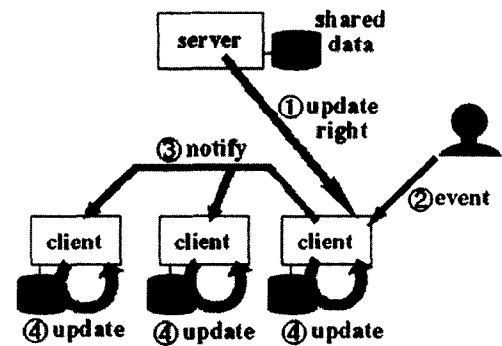




Figure 15: Client-weighted management strategy.

Table 3: Data size comparison.
(all values in byte)

Data format	Object type	vase 	bottle 
proposed method		1,402	568
VRML	Onyx2	412,338	425,283
	O2	171,821	170,899
	Indy	140,361	141,796

- [6] Hagsand,O. : Interactive multiuser VEs in the DIVE system, IEEE Multimedia, 3(1):30-39, 1996.
- [7] Heckbert, P. and Garland, M.: Survey of Polygonal Surface Simplification Algorithms, ACM SIGGRAPH'97 Course Notes, <http://www.cs.cmu.edu/~ph/mcourse97.html>.
- [8] Hoppe, H.: Progressive Meshes, Proc. ACM SIGGRAPH'96, pp.99-108, 1996.
- [9] Kado,D. et al. : Communication mechanism for shared virtual space with consideration for realtimeness and reliability, Tech. Report of IEICE, MVE98-89, 1999.
- [10] Macedonia,M.R. et al.: NPSNET: A network software architecture for large scale virtual environments, Presence, 3(4):265-287, 1994.
- [11] Nakamura,N. et al. : An internet 3-D multi-user system: Ladakh, Trans. of the IEICE, J81-D-II(5):982-991, 1998.
- [12] Nishino,H., Utsumiya,K., and Korida,K. : 3D object modeling using spatial and pictographic gestures, Proc. of the ACM VRST 1998, pp.51-58, 1998.
- [13] Nishino,H., Korida,K., and Utsumiya,K. : Deformable 3D shape representation using bimanual gestures, Trans. of IPSJ, 40(2):698-701, 1999.
- [14] Nishino,H., Utsumiya,K., Kuraoka,D., Yoshioka,K., and Korida,K. : Interactive two-handed gesture interface in 3D virtual environments, Proc. of the ACM VRST 1997, pp.1-8, 1997.
- [15] Nishino,H., Korida,K., and Utsumiya,K. : An interactive two-handed gesture interface with on-line learning facility, Trans. of the IEICE, J81-D-II(5):897-905, 1998.
- [16] Nishino,H., Mori,Y., Utsumiya,K., Yohida,K., and Korida,K. : A collaborative design framework in a distributed virtual environment, Proc. of PDPTA'99, Vol.1, pp.348-354, 1999.
- [17] Slater,M. and Wilbur,S.: A framework for immersive virtual environments (FIVE): Speculations on the role of presence in virtual environments, Presence, 6(6):603-616, 1997.
- [18] Velho, L.: Simple and Efficient Polygonization of Implicit Surfaces, Journal of Graphics Tools, 1(1):5-24, 1996.