Proceedings of DETC'97
1997 ASME Design Engineering Technical Conferences
September 14-17, 1997, Sacramento, California

DFM-97-114

HAPTIC INTERFACING FOR VIRTUAL PROTOTYPING OF MECHANICAL CAD DESIGNS

John M. Hollerbach, Elaine Cohen, William Thompson,
Rodney Freier, David Johnson, Ali Nahvi, Don Nelson, and Thomas V. Thompson II
Department of Computer Science
University of Utah
Salt Lake City, UT 84112
Email: {jmh,cohen,thompson,freier,dejohnso,nahvi,dnelson,tthompso}@cs.utah.edu

ABSTRACT

A network-based real-time control architecture has been developed which integrates a haptic interface (the Sarcos Dextrous Arm Master) with an advanced CAD modeling system (Utah's Alpha_1). New algorithms have been developed and tested for surface proximity testing, fast updates to local closest point on a surface, and smooth transitions between surfaces. The combination of these new algorithms with the haptic interface and CAD modeling system permits a user to actively touch and manipulate virtual parts as well as passively view them on a CRT screen.

INTRODUCTION

The goal of this project is to add a sense of contact and manipulation to the design process of mechanical assemblies (Hollerbach et al., 1996). There are two challenges to meeting this goal. First, mechanical CAD sytems are not designed for real-time interaction with haptic interfaces or for incorporating physics. Second, haptic interfaces have not been used to operate on complex environments or on curved surfaces, both of which are typical in virtual prototyping of mechanical assemblies. Figure 1 shows the elements of the system currently under construction to add these capabilities.

The centerpiece of the virtual world is the *Alpha_1* CAD system (Riesenfeld, 1989) running on a SGI graphics workstation (Figure 2). *Alpha_1* is a research modeling package for designing NURBS models of mechanical assemblies and has extensive manufacturing capabilities. NURBS have the advantage of compact representation, higher order continuity, and exact computation of surface normals (Snyder, 1995).

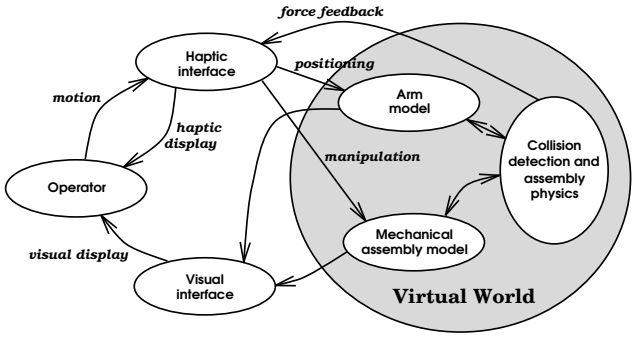

Figure 1. Real-time interaction with a CAD system requires fast NURBS evaluation techniques and assembly physics.

The haptic interface is the Sarcos Dextrous Arm Master, an advanced hydraulic force-reflecting exoskeleton (Jacobsen et al., 1990). An operator wielding the master generates position commands that are passed to both the haptic controller and the CAD system. The haptic controller uses this data to generate restoring forces and contact responses.

A virtual arm within the CAD system shadows the movements of the master arm. This virtual arm is used to determine what surfaces are needed within the haptic environment as well as aid in computation of collision detection and assembly physics. The CAD system also handles the display of the assembly and the virtual arm.

A key issue is the communication between the graphics

workstation on which the CAD model resides, and the microprocessors which control the master (Figure 2). Because of delays due to operating system limitations and Ethernet connection, it is not possible to servo directly off the CAD model. A collection of surfaces deemed to be proximal to the arm's end-effector are cached within the controller. All haptic rendering computations take place directly on the surfaces within this local environment (Thompson et al., 1997).

This paper presents how we have approached the issues of surface proximity, tracking, contact, and tracing as well as communications and haptic rendering. In this ongoing project, to date we have developed and implemented haptic surface tracing under point contact, for NURBS surfaces derived from *Alpha_1* geometric models.

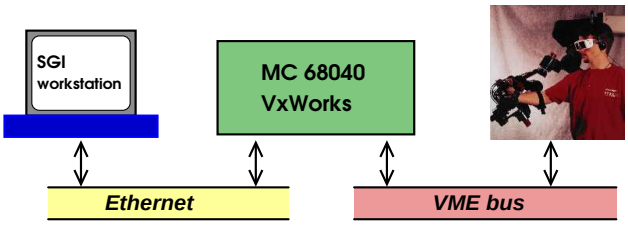

Figure 2. Real-time architecture.

BACKGROUND

In order to generate the forces necessary for realistic force feedback, a surface representation must allow rapid calculation of arm penetration depth and the corresponding surface normal. The penetration depth of the end effector into the surface is used to calculate the strength of the restoring force.

The normal of the surface above the end effector is used to give the direction of the restoring force. The penetration depth and appropriate normal can be determined by finding the closest point on the surface representation to the end effector, although other approximations could be used. In order to maintain the stiffness of the virtual surface, the force servo loop must calculate this information and response forces at several hundred Hz (Minksy et al., 1990).

Current haptic rendering systems tend to use intermediate representations when trying to render sculptured surfaces in order to maintain force servo loop rates. Adachi et al. (1995) and Mark et al. (1996) advocate the use of relatively slowly moving planar approximations when trying to represent sculptured surfaces, since contact and penetration can be quickly computed with planes. Salisbury et al. (1995) employed a polygonal representation with surface normal interpolation. However, planar representations are fundamentally limited when trying to approximate surfaces with high curvature (Mark et al., 1996). In addition, any intermediate representation must carefully deal with issues of accuracy of representation of the original surface.

The closest point on a surface S to a point E can be solved as in (Mortenson, 1985), by finding the roots of

$$(S - E) \times \left(\frac{\partial S}{\partial u} \times \frac{\partial S}{\partial v} \right) = 0. \quad (1)$$

Iterative Newton methods (Plass and Stone, 1983) can be used to solve for the roots of Eq. 1. The closest point is the nearest root of Eq. 1. However, the system may involve high degree polynomials, making the solution numerically difficult.

The CAD community has methods of tracking points on surfaces during intersection operations (Hoschek and Lasser, 1993). Many of these methods find the Euclidean closest point directly, requiring surface-plane intersections to be computed. These methods are too slow for haptic environments. Barnhill and Kersey (1990) developed a parametric marching algorithm for closest point tracking that minimizes error to a first order surface approximation. Snyder (1995) uses Newton iteration to improve an approximation to the closest point between two surfaces during collision detection, as does Baraff (1990). Lin and Manocha (1995) use a polyhedral first pass along with a global closest point solver to initiate closest point tracking between two surfaces and local methods for fast updates.

Current numerical methods may take several iterations, and thus several surface evaluations to converge. In addition, they are sensitive to the parameterization of the surface.

SURFACE INTERACTIONS

A key issue is that, because of network and workstation latencies, it is not possible for the master to servo directly off of the CAGD models on the workstation. Instead, we need a surface representation which can be efficiently communicated to and stored on the micros, and for which the micros can determine end effector penetration and restoring force direction during a collision.

To avoid the difficulties associated with intermediate representations, we servo off of the parametric representation directly, and use a novel tracing method (Thompson et al., 1997) to find the approximate closest point on the surface and its associated normal. As a basis for comparision,

we also implement an intermediate representation that consists of a set of surface points and their associated normals. Both methods share a similar computation cycle as follows.

1. The workstation receives periodic updates about master position from the micros over the network. The workstation performs a general point-to-surface calculation to isolate the closest surface portions. A surface patch is extracted and sent over the network to be cached with the micros as necessary (so the virtual slave position doesn't go off the edge of the local representation).
2. The micros perform collision detection on the surface patches, and calculate forces for reflection back to the master.

Next we provide details on surface proximity testing, patch extraction, the local geometry representation, and collision detection.

Surface Proximity Testing

For each method, representations of geometry near the master's location are periodically sent from the workstation to the micro. The workstation must determine when surfaces are in near enough proximity to the master's location to be sent to the micro, and what portion of the model must be sent.

As a rough check for surface proximity, we use bounding boxes around each surface (Johnson and Cohen, 1997). Distance from a point to a bounding box is a trivial computation, and we can quickly discard the majority of the surfaces in the environment.

For the remaining surfaces, we use a method we refer to as "nodal mapping" (Snyder, 1995) to find a first order approximation to the closest point on the surface. First, we project the end effector point onto the control mesh of the NURBS surface. Each vertex of the control polygon has an associated (u, v) parametric value (Cohen and Schumaker, 1985), so by interpolating between vertex values using the barycentric coordinates of the projection, we determine an approximate (u, v) for the projected point.

Parametric Tracking

When a surface becomes active, the micros use the parametric value of the approximate closest point, found through nodal mapping, to initialize a local closest point tracking method. This tracking method works directly on a parametric surface and runs at several hundred HZ on the 68040 microprocessors, making it suitable for direct haptic rendering of a surface (Thompson et al., 1997).

We present the local closest point tracking method on a parametric curve, $\gamma(u)$, for simplicity. The closest point

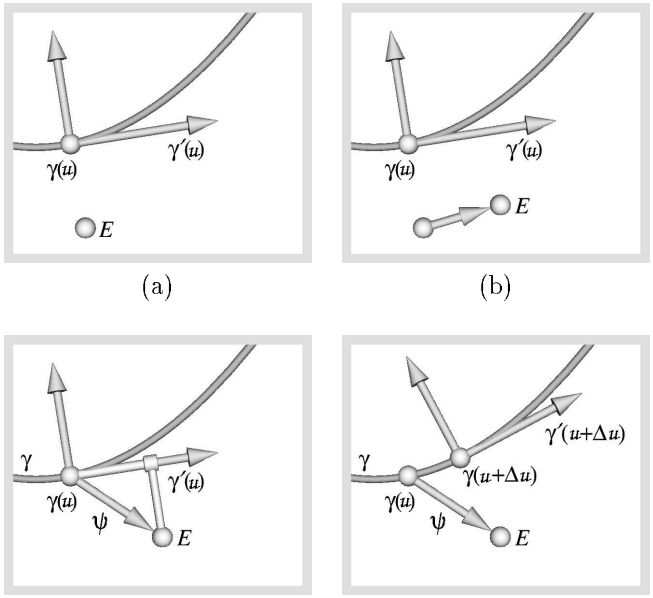

Figure 3. (a) Initial state. (b) End-effector moves. (c) Projection of position onto surface tangent plane. (d) New surface point and tangent plane found via parametric projection.

method uses the previous point on the curve $\gamma(u)$, the tangent vector at $\gamma(u)$, $\gamma'(u)$, and the current end-effector location, E, to determine a new approximate closest point on the curve (Figure 3).

The parametric velocity, $\gamma'(u)$, relates changes in position along the curve in Euclidean space to changes in position in parametric space (Eq. 2).

$$\gamma'(u) = \frac{d\gamma}{du} \approx \frac{\Delta\gamma}{\Delta u}. \tag{2}$$

We can approximate $\Delta\gamma$ as the projection of the movement vector onto the curve tangent at $\gamma(u)$ (Figure 3c). The curve parametric velocity over the range of movement is found using a first order approximation, the value $\gamma'(u)$. The value of Δu can now be found.

The velocity can be efficiently found using only the values of the control mesh, the curve knot vector, and the curve order. This efficient method applies for points on the curve where there are $k-1$ knots at index i^* with the value u^*. Since $k-1$ knots were inserted at u^* to create an evaluation point, this condition holds for closest point tracking. The resulting equation for $\gamma'(u)$ is

$$\gamma'(u^*) = \frac{(k-1)}{u_{i^*+k} - u_{i^*+1}}(P_{i^*+1} - P_{i^*}). \quad (3)$$

and the equation for Δu reduces to

$$\Delta u \approx \frac{\langle\, \psi\ ,\ (P_{i^*+1} - P_{i^*})\,\rangle}{\|P_{i^*+1} - P_{i^*}\|^2} \left(\frac{u_{i^*+k} - u_{i^*+1}}{k-1}\right). \quad (4)$$

For surfaces, the method is essentially the same, although the projection step now requires projection onto the tangent plane, $S'(u, v)$, of the surface. Barycentric coordinates are used to derive Δu and Δv. Our implementation of the tracking method, when used to trace a single surface, runs at 1400Hz on the Motorola 68040 processor that is used in the haptic process.

Contact and Tracing

The haptic loop determines surface contact and resulting arm forces. Contact is initiated when the penetration depth becomes larger than zero. The penetration depth is calculated by projecting the arm location onto the surface normal. During contact, the local closest point is updated as during the tracking phase.

Point Set Intermediate Representation

The point set representation uses the approximate closest point as a center point in a mesh of surface points evaluated at fixed parametric intervals (Figure 4). This surface evaluation is efficiently done through multiple knot insertions at the desired parametric locations using the Oslo algorithm (Cohen et al., 1980). The surface normals can also be extracted from the refined surface (Riesenfeld et al, 1981). These points and normals are used to represent the local geometry in the micro.

With the point set representation, we first find the closest point in the point set to the end effector. Penetration depth is determined by projecting the end effector onto the closest point's normal. The point normal is used to determine the direction of the restoring force.

NETWORKING

Because the real-time interaction of the arm with the virtual surface is critical, networking design and throughput are very important. The general guidelines followed in the project are:

- UDP is faster than TCP, and the associated packet loss is acceptable for state update that occurs continuously.

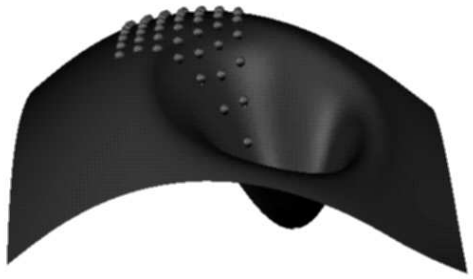

Figure 4. A 7x7 point set superimposed over a NURBS surface.

- TCP is needed for guaranteed update of surfaces or state changes that are sent only once.
- Surface updates are not always necessary.
- Parallelism of multiple processors in our environment helps by having dedicated "network" processors.

In our implementation, it was found necessary to utilize two microprocessor boards. One is responsible for all communications, while the other implements the force reflection loop. The two boards exchange information through shared memory.

Microprocessor Real-Time Architecture

We employ the ControlShell (Real Time Innovations, Inc., Sunnyvale, CA) object-oriented real-time software package for controlling the Master and for network communication. ControlShell runs on top of the VxWorks real-time kernel and development environment (Wind Rivers, Inc.) for the microprocessor system. Network communication has been implemented as ControlShell objects.

Point Set Networking

The ControlShell force loop needs an updated mesh whenever the operator has moved the end effector to a new location. To minimize the amount of slower TCP communication overloading the ethernet, regular broadcasts via UDP of the position is sent by the networking processor at one-tenth the rate of the "force control" processor. The SGI workstation responds to a large change in position (approximately 1 cm) by responding with a new mesh.

This asynchronous style of communication saves network bandwidth, and uses shared memory to send a new mesh quickly to the force loop. Although meshes are sent on demand, the system is capable of transmitting meshes at a 100 Hz rate, which is quite sufficient in maintaining a closest mesh to the arm user's hand.

Parametric Surface Networking

The requirements of this version of the surface representation are similar to that of the point set version. However, in that version a dense point was sent at high rates to the controller. This made for an abundance of network traffic. Making matters worse, no truly effective caching is possible for two reasons. First, the data has no connectivity, so the cache cannot determine when to use or flush cached points. Second, the number of points needed to cover a region with sufficient fidelity is so high that little space is left to store any meaningful number of cached points.

In the parametric network version these limitations are turned into strengths. First, the data represents an entire continous region, so there is no need to worry about overlap and ordering. Second, and perhaps most importantly, the representation is much more compact. With less data needed to represent a large area of the workspace, the real-time system can cache several patches and significantly reduce the network traffic.

The SGI workstation receives regular position updates from the real-time system via UDP packets. When a surface patch is deemed close for the first time, the workstation sends this surface patch to the real-time system's dedicated networking board via TCP/IP. If a patch has already been cached on the micro, the workstation sends a small TCP/IP packet to the micro notifying that that surface is now active. Correspondingly, a deactivation packet is sent if the workstation determines that a particular surface is no longer in the focus. To avoid "thrashing" of active and deactive states, a small buffer of 10 cm is used.

Shared memory update to the force control board is somewhat complex as only one activate, deactivate, or surface patch should be sent to this board at a time. Furthermore, it is vital that none of this data be ignored as both sides of the system must remain in sync. The details reduce to the "Producers-Consumers" problem, making use of mutual exclusion, "who's turn" update, and the TCP/IP buffer as the queue of requests.

DYNAMICAL INTERACTIONS

At the moment, our dynamics modeling is limited to surface interaction. A key issue is the choice of a compliance model or a stiffness model for the interaction of the Master with a virtual assembly (Yoshikawa and Ueda, 1996). We have chosen a stiffness model, where the user generates positions against the virtual assembly and the simulation returns forces. This approach is well suited for the Sarcos Master, which is a better force source than a position source. The user displacements are incorporated into an inverse dynamics computation. Geometric violations are accomodated by restoring forces.

Surface Model

How one models a surface for contact and restoring forces is apparently far from straightforward. Although we began with the usual viscoelastic surface models, we have subsequently adopted the nonlinear viscosity model of (Marhefka and Orin, 1996) for energy conservation:

$$f = -(\lambda x^n)\dot{x} - k x^n \tag{5}$$

where f is the normal force, x the normal displacement, λ the viscosity coefficent, and k the stiffness. From experience we find that $n = 0.5$ provides a good response. Furthermore, we have adopted the force impulse model of (Ellis et al., 1996), which avoids active surfaces (energy gained during collisions) due to lag introduced by discretization.

Stick-Slip Friction

Aside from normal forces, adding tangential forces due to friction is properly required for realistic interactions. Salcudean and Vlaar (Salcudean and Vlaar, 1994) presented a stick-slip model, which employs a velocity threshold to define sticking but which only employs viscous friction during slipping. We have modified their model by incorporating both dynamic and static coefficients of friction in the model. Assume the haptic interface is moving against a surface. In the *slip* phase, the friction force $\mathbf{f}_f$ is

$$\mathbf{f}_f = \mu_d ||\mathbf{n}|| \frac{\mathbf{v}}{||\mathbf{v}||} \qquad \text{if } ||\mathbf{v}|| > v_{min} \tag{6}$$

where μ_d is dynamic coefficient of friction, $\mathbf{n}$ is the normal force, $\mathbf{v}$ is the tangential velocity of the haptic interface, and v_{min} is a small threshold velocity. As soon as $||\mathbf{v}||$ becomes smaller than v_{min}, the *stick* phase begins.

Figure 5 shows the transition from the *slip* to *stick* phase. Point $\mathbf{a}$ represents the transition point where the velocity reaches a threshold of v_{min}. During this transition, a virtual spring is formed whose stiffness $k_f||\mathbf{n}||$ is proportional to the normal force. It pulls the haptic interface towards $\mathbf{c}$ (stick center). The spring force at this moment is:

$$k_f ||\mathbf{n}|| \, ||\mathbf{a}\text{-}\mathbf{c}|| = \mu_d ||\mathbf{n}|| \tag{7}$$

As long as the spring force is less than the static friction limit $\mu_s||\mathbf{n}||$, the *stick* phase holds. In other words, haptic interface is trapped in a sphere of radius $\mu_s/\mu_d||\mathbf{a}\text{-}\mathbf{c}||$ centered at $\mathbf{c}$. $\mathbf{c}$ is found by:

$$\mathbf{c} = \mathbf{a} - \frac{\mu_d ||\mathbf{n}||}{k_f ||\mathbf{n}||} \frac{\mathbf{v}}{||\mathbf{v}||} = \mathbf{a} - \frac{\mu_d}{k_f} \frac{\mathbf{v}}{||\mathbf{v}||} \tag{8}$$

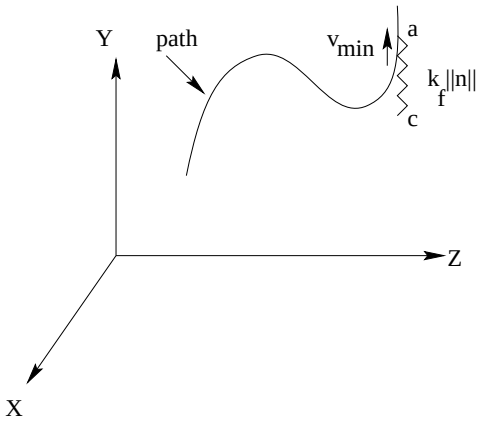

Figure 5. Transition from *slip* to *stick* phase.

where **v** is the velocity just before transition. It can be easily shown that the maximum stick radius is μ_s/k_f. The harder the contact materials of surface and haptic interface, the bigger k_f. The *slip* phase begins when the haptic interface is μ_s/k_f, or equally $\mu_s/\mu_d||\mathbf{a}\text{-}\mathbf{c}||$ away from **c**. We can imagine that the virtual spring has a rupture limit of $\mu_s||\mathbf{n}||$.

Master

The Sarcos Dextrous Arm Master is presently driven by the Advanced Joint Controller (AJC), a proprietary Sarcos system with cards that implement an analog controller, analog-to-digital conversion of joint position and torque sensing, digital-to-analog output to electrohydraulic servovalve drivers, digitally writable gains, feedforward control, and communication with the VMEBus via a parallel port. The Motorola 68040 microprocessors on the VMEBus interact with the AJC via the parallel port, and with workstations over an ethernet connection.

Our own software modifications include new calibration and gravity compensation procedures, which are essential for proper functioning of the arm and for removing the weight of the Master from the operator (Ma and Hollerbach, 1996).

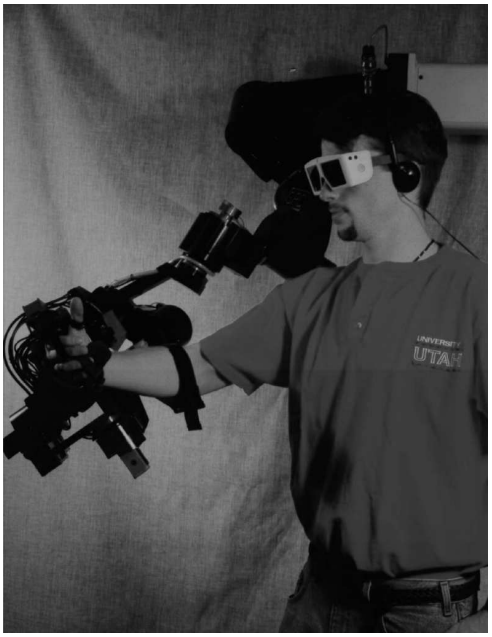

Figure 6. Sarcos Dextrous Arm Master.

Figure 7. A goblet is traced by the master (represented as a small sphere).

RESULTS

We have successfully traced a variety of sculptured surface models, including a bumpy surface, a cylinder, and a goblet. The goblet in Figure 7 illustrates the capability of the parametric tracing algorithm to trace complex multi-surface models. The master position is represented by the small sphere.

We compared the performance of the mesh-based versus parametric-based surface patch representation, with respect to computational load, depth penetration and variance. The mesh networking board was more heavily loaded due to the mesh updates, but was still able to keep up with its force loop board at 333 Hz. The bottleneck for the parametric version was the force loop board, which attained servo rates of 250 Hz for multiple surface tracking and on-board penetration depth and force computation. Nearly zero penetration is possible with very high surface response gains, but those gains resulted in an untraceable surface due to the large bounce and fact that the arm cannot achieve

infinite stiffness.

The more accurate normals and surface points for the parametric evaluation resulted in nicer depth penetration characteristics as shown in Table 1. The tests were run

Model	Method	Sample Points	Depth Max	Depth Ave
Flat	mesh	832	4.646	1.852
	DPT	1450	2.246	0.837
Bumpy	mesh	699	6.256	1.515
	DPT	2030	1.894	0.661
Cylinder	mesh	2541	13.653	1.787
	DPT	694	2.033	0.629
Goblet	DPT	1125	1.365	0.336

Table 1. Average and maximum penetration depth for mesh vs. direct parametric tracing of different models. Sample points is the number of distinct contact evaluations and all depths are given in centimeters.

with identical proportional and derivative response gains. A little more than a centimeter penetration occured on average during the surface tracing, on surfaces that roughly filled the $1m^3$ arm working volume. Note that the averages for the parametric surface patch method were much lower.

DISCUSSION

This paper has presented an approach towards incorporating a haptic interface into a mechanical CAD system, in order to provide the user with a sense of touch as well as of sight. The interaction between haptic interface and CAD system imposes stringent real-time requirements on collision detection and dynamical simulation to generate contact forces and virtual objection motions.

Our approach involves caching surface patches of a full geometric model, residing on a workstation, on a microprocessor system, which must interact with the master at a high rate. Communication between the micros and the workstation occur at a lower rate. From periodic master position updates sent over the network from the micros, the workstation extracts a surface patch to be transmitted to the micro, or activates one of the patches already cached on the micro.

Two surface patch representations were examined. It was found that direct operation on a parametric surface patch representation was feasible, and preferable to an intermediate representation approach involving a mesh. Methods for real-time computation of proximity and contact utilizing parametric surface patches were presented.

The first implementation has concentrated on surface tracing tasks under point contact. We are presently working on surface-to-surface interactions, such as a grasped object interacting with another, and on adding relative motion between CAD objects, such as turning a crank or gear train. Some specific components are

- Support for moving surfaces.
- Introduction of trimmed NURBS models as are common in mechanical design.
- Use of real-time surface-surface closest point and collision detection (Johnson and Cohen, 1997).
- Use of sound as additional feedback.
- Assembly data structures for verification and assembly analysis.
- Wall models to represent surface properties.
- Immersive environments for design.

ACKNOWLEDGMENT

Support for this research was provided by NSF Grant MIP-9420352, by DARPA grant F33615-96-C-5621, and by NSERC NCE IRIS Project HMI-8.

REFERENCES

Adachi, Y., Kumano, T., and Ogino, K., 1995, "Intermediate representation for stiff virtual objects," *Proc. Virtual Reality Annual Intl. Symp.*, Research Triangle Park, NC, Mar. 11-15, pp. 203-210.

Baraff, D., 1990, "Curved Surfaces And Coherence For Non-penetrating Rigid Body Simulation," *Proc. SIGGRAPH 90*, Dallas, August 6-10, pp. 19-28.

Barnhill, R.E., and Kersey, S.N., 1990, "A Marching Method For Parametric Surface/Surface Intersection," *Computer Aided Geometric Design*, 7, pp. 257-280.

Cohen, E., Lyche, T., and Riesenfeld, R., 1980, "Discrete B-Splines and subdivision techniques in computer aided geometric design and computer graphics," *Computer Graphics and Image Processing*, 14 no. 2, .

Cohen, E., and Schumaker, L., 1985, "Rates of convergence of control polygons," *Computer Aided Geometric Design*, 2 no. 1-3, .

Ellis, R.E., Sarkar, N., and Jenkins, M.A., 1996, "Numerical methods for the haptic presentation of contact: theory, simulations, and experiments," *Proc. ASME IMECE Haptics Symp.*, in press.

Hollerbach, J.M., Cohen, E., Thompson, W.B., and Jacobsen, S.C., "Rapid virtual prototyping of mechanical assemblies," *Proc. 1996 NSF Design and Manufacturing Grantees Conf.,* Albuquerque, Jan. 2-5, 1996, pp. 477-478.

Hoschek, J., and Lasser, D., 1993, *Fundamentals of Computer Aided Geometric Design*, A K Peters.

Jacobsen, S.C., Smith, F.M., Iversen, E.K., and Backman, D.K., 1990, "High performance, high dexterity, force reflective teleoperator," *Proc. 38th Conf. Remote Systems Technology*, Washington, DC, Nov., pp. 180-185.

Johnson, D.E., and Cohen, E., "Minimum Distance Queries For Polygonal and Parametric Models," Technical Report UUCS-97-003, University of Utah, Department of Computer Science, Feb. 26, 1997.

Lin, M., and Manocha, D., 1995, "Fast Interference Detection Between Geometric Models," *The Visual Computer*, pp. 542-561.

Ma, D., and Hollerbach, J.M., 1996, "Identifying mass parameters for gravity compensation and automatic torque sensor calibration," *IEEE Intl. Conf. Robotics and Automation*, Minneapolis, April 21-28, pp. 661-666.

Marhefka, D.W., and Orin, D.E., 1996, "Simulation of contact using a nonlinear damping model," *IEEE Intl. Conf. Robotics and Automation*, Minneapolis, April 21-28, pp. 1662-1668.

Mark, W.R., Randolph, S.C., Finch, M., Van Verth, J.M., and Taylor III, R.M., 1996, "Adding force feedback to graphics systems: issues and solutions," *Computer Graphics (Proc. SIGGRAPH)*, New Orleans, Aug. 4-9, pp. 447-452.

Minsky, M., Ouh-Young, M., Steele, M., Brooks, F.P. Jr., Behensky, M., 1990, "Feeling And Seeing: Issues In Force Display," *Proc. Symposium on Interactive 3D Graphics*, Snowbird, Utah, pp. 235-243.

Mortenson, M., 1985, *Geometric Modeling*, John Wiley & Sons.

Plass, M., and Stone, M., 1983, "Curve-Fitting With Piecewise Parametric Cubics," *Proc. SIGGRAPH*, Detroit, July 25-29, pp. 229-236.

Riesenfeld, R., Cohen, E., Fish, R., Thomas, S., Cobb, E., Barsky, B., Schweitzer, D., and Lane, J., 1981, "Using the Oslo algorithm as a basis for CAD/CAM geometric modelling," *Proc. National Computer Graphics Association.*

Riesenfeld, R., 1989, "Design tools for shaping spline models," *Mathematical Methods in Computer Aided Geometric Design*, edited by T. Lyche and L. Schumaker, Academic Press.

Salcudean, S.E., and Vlaar, T.D., 1994, "On the emulation of stiff walls and static friction with a magnetically levitated input/output device," *Proc. ASME Dynamic Systems and Control Division*, Chicago.

Salisbury, K., Brock, D., Massie, T., Swarup, N., and Zilles, C., 1995, "Haptic rendering: programming touch interaction with virtual objects," *Symp. on Interactive 3D Graphics*, Monterey, CA, pp. 123-130.

Snyder, J., August. 6-11, 1995, "An Interactive Tool For Placing Curved Surfaces Without Interpenetration," *Proc. SIGGRAPH 95*, Los Angeles, pp. 209-218.

Thompson II, T.V., Johnson, D.E., Cohen, E.C., April 27-30, 1997, "Direct Haptic Rendering Of Sculptured Models," *Proc. Symposium on Interactive 3D Graphics*, Providence, RI, pp. 167-176.

Yoshikawa, T., and Ueda, H., 1996, "Construction of virtual world using dynamic modules and interaction modules," *IEEE Intl. Conf. Robotics and Automation*, Minneapolis, April 21-28, pp. 2358-2364.