Proceedings of Symposium on Haptic Interfaces
ASME International ME'99 Congress and Exposition
November 14-19, 1999, Nashville, Tennessee

DIRECT HAPTIC RENDERING OF COMPLEX TRIMMED NURBS MODELS

Thomas V Thompson II **Elaine Cohen**
Department of Computer Science
University of Utah

ABSTRACT

The most accurate haptic rendition of a virtual model is produced when the haptic algorithm acts directly on the actual model and not an intermediate representation. In the modeling and design communities the de facto model representation standard is NURBS. Further, more powerful systems provide trimming and adjacency information within the models representation. This additional information permits more complex models to be expressed succinctly but also increases the complexity of representation. In this paper we present an algorithm that supports direct haptic rendering of models constructed from trimmed NURBS surfaces. Our distributed system links an advanced modeling system to a force-reflecting device. In addition, we present extensions to the algorithm which support model manipulation, dimensioned probes, and multi-probe contact.

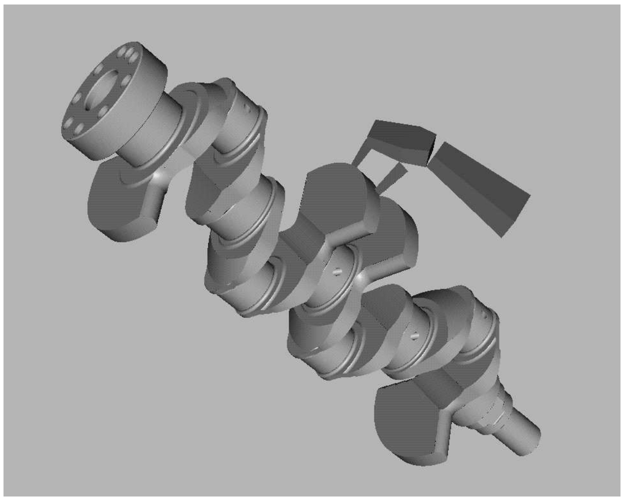


Figure 1. Multiple endpoint haptic rendering of a four cylinder crank shaft constructed from trimmed NURBS surfaces.

1 INTRODUCTION

The passive graphical display of complex models can convey only limited visual information. Even with the array of presentation options supplied to a user by modeling packages, such as isoline drawings, shaded images, and animations, the designer is often left without information that could easily be gathered if the model could be interrogated by touch (Gibson, 1966). Haptic rendering supplies this channel of feedback to the user by simulating the forces generated by contact with, and surface tracing of, a virtual model (Fig. 1). This increased level of interaction facilitates a greater understanding of complex models and adds to the sense of realism in virtual environments (Hollerbach et al., 1996; Stewart et al., 1997).

While the graphical display of a model is almost exclusively accomplished by first converting it to a collection of polygons, the model itself often starts with a different geometric representation. In fact, a master CAD model is almost always described by NURBS throughout its life (Piegl and Tiller, 1995). Being a parametric representation, NURBS surfaces have the advantage of compactness, higher order continuity, and exact computation of surface tangents and normals. All of these properties are useful in complex, realistic virtual environments (Snyder, 1995).

The ability to trim away arbitrary portions of a NURBS surface and define adjacencies under boolean set operations is a powerful extension to any modeling package. These constructive solid geometry results form a large class of models which can be much more readily and succinctly expressed using trimmed NURBS than non-trimmed NURBS. Furthermore, the use of trimmed NURBS is fairly widespread, making it important for a haptic rendering algorithm to handle them. This increase in expressiveness does not come without a cost. Since the trimming curve (or curves) representing the boundary of a surface is frequently expressed as a piecewise linear curve with a high number of segments, determining when haptic contact is to transition from one surface to another can be a complex task. However, once such a transition is detected, topological adjacency information can allow for efficient computation of the exact transition point.

A major contribution of this paper is that we use a model-oriented approach which enables an environment populated by a broad class of models to be haptically rendered (Fig. 1). Enabling the user to trace directly on the actual CAD model instead of an intermediate or alternate representation achieves the most accurate haptic rendering results. To this end, we introduce direct haptic rendering of complex models constructed from trimmed NURBS surfaces. In order to realize this approach, we have developed and tested algorithms for model proximity testing, fast update of global and local closest point approximations, computationally efficient surface evaluation and normal evaluation techniques, and smooth efficient transitioning across trimmed surface boundaries. Further, we present extensions including model manipulation, dimensioned probes, and multiple probe contact. These algorithmic results are tested within a complete system that integrates a research modeling package, Alpha_1 (Riesenfeld, 1989; Riesenfeld, 1993) with both a Sarcos Dextrous Arm Master (Jacobsen et al., 1990) and a Phantom haptic feedback device (Massie and Salisbury, 1994).

2 BACKGROUND

The goal of a haptic rendering system is to generate forces that can be applied to a user's hand or arm to accurately produce a sense of contact with a virtual model. These forces, called restoring forces, resist penetration into the virtual model and are calculated using a wall model. These response models often have a restoring force proportional to the penetration depth (Colgate and Brown, 1994) and in the direction of the surface normal at the contact point, typically a local closest point. An accurate force computation depends on a good tracking and tracing algorithm since the magnitude and direction of the force depends on the accuracy of the calculated closest point and normal. Importantly, the force servo loop, and hence the closest point calculation, must run at several hundred Hz in order to maintain stiff virtual surfaces (Minsky et al., 1990). This high update rate limits the complexity of the algorithms for finding the closest point and thereby has dictated the types of models that could be rendered.

For instance, Zilles and Salisbury (1995) advocate using a constraint-based system to trace polygonal models. These systems are often limited to simple models since the high polygon count of complex models requires too much processing time. Ruspini et al. (1997) extend this work to handle larger polygon counts and to permit more general graphics primitives, such as points and lines, to be traced by a dimensioned probe.

Adachi (1993) employs distribution functions and Salisbury and Tarr (1997) use implicit surfaces to permit quality tracing of sculptured models. However, the design and graphics communities by and large model using NURBS. As such, to use these methods requires a conversion from the NURBS model into one of these other representations. This conversion is not only a complicated task but can be numerically unstable resulting in inaccurate models defined by high order functions.

Others (Adachi et al., 1995; Mark et al., 1996) propose using intermediate representations to simplify the tracing of sculptured models. These systems are bound by network limitations, and therefore updates are slow. While this approach permits the tracing of complex models, it suffers from poor temporal performance and cannot accurately reproduce surfaces with high curvature.

Thompson et al. (1997a) as well as Johnson and Cohen (1998) have demonstrated direct haptic rendering of sculptured models constructed from untrimmed NURBS surfaces using direct parametric tracing. While sculptured models can be traced with quality results using these methods, they are predominantly surface oriented approaches since the models contain no topological adjacency information. The method requires tracking a closest point on each proximal surface to allow tracing across surface boundaries. This in turn limits both the complexity and number of models populating the environment.

Nelson et al. (1999) provide a closed form solution in the velocity domain for tracing sculptured NURBS models using a locally convex end-effector model. While this method allows the end-effector to be sculptured, the contact model is a single point and the approach is surface oriented with no transitioning algorithm.

3 SYSTEM OVERVIEW

A haptic virtual environment must meet the update rate constraints of the visual and haptic displays. While it may be acceptable for the visual display to update at twenty frames per second, the haptic display must maintain rates of several hundred Hz for the virtual surfaces to feel solid. Both the visual and haptic display individually tax a system, therefore, we distribute the *simulation process* and *haptic process* onto different special purpose computers (Thompson et al., 1997a; Thompson et al., 1997b).

The modeling environment runs within the simulation process on a graphics workstation which handles the visual display and performs computations typically global in scope. When the simulation process detects an appropriate global event, such as the proximity of the probe to a model, it signals the haptic process, which continues the computation with low-latency, local methods. The haptic process runs on real-time microcomputer boards and performs computations which are typically local in scope.

Several Ethernet channels connect the two processes to facilitate the various forms of communication in our system. Both processes have a model manager whose sole function is to maintain data consistency across the system. Data caching on both sides of the system, in conjunction with the distinct division of labor between the two processes, keeps the communication overhead at a minimum. The combination of independent processes maximizing computational capacity, low communication overhead, and synchronized visual and haptic display creates a realistic haptic virtual environment.

4 TRIMMED NURBS MODELS

Non-Uniform Rational B-Spline (NURBS) surfaces are highly compact and yet very expressive as a representation for modeling. A NURBS surface is a bivariate vector-valued piecewise rational function of the form

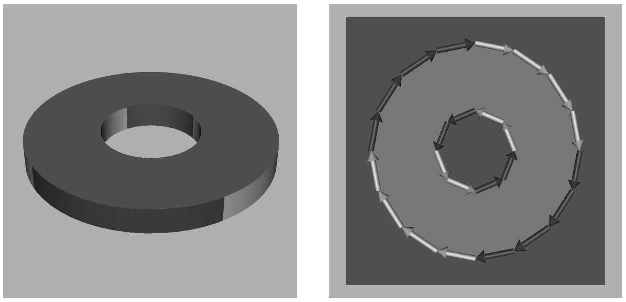
$$S(u,v) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} P_{i,j} w_{i,j} B_{j,k_v}(v) B_{i,k_u}(u)}{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{i,j} B_{j,k_v}(v) B_{i,k_u}(u)}, \qquad (1)$$

where the $\{P_{i,j}\}$ form the control mesh, the $\{w_{i,j}\}$ are the weights, and the $\{B_{i,k_u}\}$ and $\{B_{j,k_v}\}$ are the basis functions defined on the knot vectors $\{u\}$ and $\{v\}$ for a surface of order k_u in the u direction and k_v in the v direction.

The various properties of a NURBS surface, including a local convex hull property, and the ability to evaluate surface points, normals and tangents, along with its intuitive control characteristics make it a good representation for modeling and design. These properties have led to NURBS becoming the *de facto* industry standard for the representation and data exchange of geometric models (Piegl and Tiller, 1995).

Trimmed NURBS models are constructed by cutting away portions of a NURBS surface using trimming curves in parametric space. In our system trimming information is represented as directed closed polygons called *trimming loops*. Each individual linear portion of the loop is called a *segment*. A collection of connected segments that represents shared boundary between two surfaces is referred to as an *edge*. Portions of the surface domain to the left of a loop are considered cut-away while pieces to the right are deemed part of the model. Note that each surface that is part of a model contains at least one trimming loop. If there is no portion of the surface being cut away then this loop simply surrounds the domain of the surface.

Consider the model in Fig. 2a of a simple disc with a hole cut through it. The top surface will have two trimming loops within its parametric domain as shown in Fig. 2b. Notice that the direction of the two loops indicate that the dark regions are to be cut away. The outer clockwise loop cuts away the outermost region while the inner counterclockwise loop cuts away the center region representing the hole. The edges in Fig. 2b are illustrated in alternating brightness to indicate the patch in Fig. 2a that is adjacent to each edge.



(a) (b)

Figure 2. A trimmed NURBS model (a) and its parametric domain containing the trimming loop information (b).

5 DIRECT HAPTIC RENDERING

The goal of direct haptic rendering is to enable the user to feel the actual designed model instead of some secondary representation. In addition to an enhanced tracing experience, using the actual model also allows the designer to modify the model without having to wait for the haptic system to convert the model in a time-consuming preprocessing step. The difficult part of this approach lies in the evaluation of the model to acquire the requisite closest point and surface normal needed by the force response model. These calculations must complete in a short enough time to permit the entire haptic process to maintain its high update rate. Also, the method must produce what we refer to as a *local closest point* since tracing using a global closest point results in several complications and, in fact, potentially erroneous results (Thompson et al., 1997a).

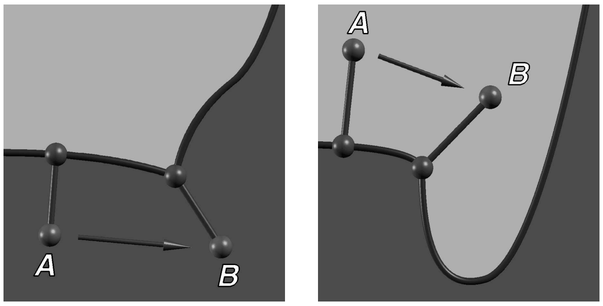
The act of haptically tracing a virtual model can be broken down into several phases. The system checks each model for proximity to the probe and activates those models deemed proximal. The tracking algorithm then tracks a closest point on the model until contact is made, at which point the tracing algorithm takes over. While in contact, the tracing algorithm maps the movement of the probe to movement along the surface of the model. Transitioning is indicated if the probe's movement causes the local closest point to cross a trimming loop boundary.

5.1 Proximity Detection

Our algorithm needs to trace only a single point per model per end-effector. However, it is still advantageous to keep the number of models active to a minimum. Doing so saves processor time and maintains high update rates. To this end, our system checks the proximity of the probe as it moves throughout the environment to each model. We use a first order approximation to the closest point found by a method referred to as nodal mapping (Thompson et al., 1997a). The simulation process performs this computation since it is a global computation that requires the processing power of the workstation. When the probe moves close enough to potentially contact a model, an activation packet is sent to the haptic process indicating that a closest point should be tracked. This packet contains the ID for the model being activated along with the surface ID and parametric location of the closest point. The parametric value then seeds the tracking algorithm.

5.2 Tracking

Any movement near an active model, while not in contact with that model, is referred to as tracking. Contrary to tracing, the closest point for tracking must be the global closest point. This ensures proper contact detection by permitting the closest point to jump across concavities and to climb convex regions. For example, Fig. 3 demonstrates the different closest point requirements. During tracing, the probe moves from position A to position B resulting in the closest point becoming bound to the intersection of two surfaces (Fig. 3a). This is the correct response: the probe is trying to move inside a second surface and should be restricted from movement in that direction. If the same algorithm was used for tracking, a similar scenario could occur (Fig. 3b). Again the probe moves from position A to position B, which results in the tracked point being bound to an edge. In this case, however, the closest point should not be bound. In fact, if the probe were to continue along the current path and intersect the model, the contact would not be detected since the penetration would not occur at the location of the bound tracked point.



(a) (b)

Figure 3. A bound closest point is desired when in contact and tracing (a) but when tracking while not in contact (b) it is not.

The tracing algorithm cannot be used exclusively for tracking since it assumes the probe is in contact and therefore the next desirable closest point should correlate to a movement bound to the surface of the model. In the tracing algorithm, it is appropriate for the closest point to hold its position since its purpose is to restrict the probe's movement in certain directions. During tracking, however, there should be no restrictions on the probe's movement or the closest point which shadows it.

Currently, it is not feasible to determine the global closest point to a trimmed NURBS model at haptic rates. Therefore, we use a hybrid approach where the tracing algorithm is augmented by periodic re-seeding with the global closest point. The simulation process calculates an approximate global closest point using an algorithm based on a time-critical method that spends less time on objects outside the region of interest (Johnson and Cohen, 1997). The haptic process accepts this point and uses it to re-seed the tracing algorithm. This periodic re-seeding allows the tracing algorithm to form a good approximation to the global closest point between updates. We have found this to be very effective since the probe typically does not move very far between updates. Even though contact detection may be delayed for several cycles, the cycle rate is so high that the delay is, in practice, imperceptible to the user.

5.3 Contact and Tracing

At the heart of every haptic system is its ability to detect the contact of the probe with a virtual model and to simulate the act of tracing along the surface of the model. Contact occurs when the probe first intersects the virtual model, resulting in a positive penetration depth. Once contact has been established, any lateral movement along the surface indicates tracing. As the probe moves, a local closest point on the surface of the model is found that shadows the probe's movement. The accuracy of this closest point and its associated normal is essential since the restoring force calculation depends directly upon this result.

We relate movement of the probe to movement along the surface using direct parametric tracing (Thompson et al., 1997a). The algorithm has been shown to run at interactive rates and produce accurate results making it suitable for direct haptic rendering. The algorithm is seeded with the surface evaluation point $S(u^*, v^*)$, calculated using refinement (Cohen et al., 1980) where (u^*, v^*) are the parametric coordinates for the point of contact.

Movement along a surface in Euclidean space relates to movement in parametric space by the partial derivatives of the surface

$$\frac{\partial S}{\partial u} \approx \frac{\Delta S_u}{\Delta u} \;\; , \;\; \frac{\partial S}{\partial v} \approx \frac{\Delta S_v}{\Delta v},$$

where ΔS_u and ΔS_v are the change on the surface along the u and v isocurves respectively. A good approximation for ΔS_u and ΔS_v is the projection of the probe onto the surface tangent plane. The coordinates of this projection within the tangent plane are used to derive Δu and Δv.

A key element of the algorithm is the efficient computation of the surface tangents $\frac{\partial S}{\partial u}$ and $\frac{\partial S}{\partial v}$. The calculation of $S(u^*, v^*)$ by refinement results in the new knot vectors $\{\hat{u}\}$ and $\{\hat{v}\}$, the new weights $\{\hat{w}_i\}$ and a new control mesh $\{\hat{P}_{i,j}\}$ where $\hat{P}_{i^*,j^*} = S(u^*, v^*)$. Furthermore, the i^* column and the j^* row in the control mesh form control polygons for isocurves that pass through the point $\hat{P}_{i^*,j^*}$. This allows the calculation of the two tangent vectors to be performed using the more simple curve equation. The isocurve defined at v^* in the j^* row of the control mesh is given by

$$\gamma_{j^*}(u) = \frac{\sum_{i=0}^{m} \hat{P}_i \hat{w}_i B_{i,k_u}(u)}{\sum_{j=0}^{m} \hat{w}_j B_{j,k_u}(u)}, \tag{2}$$

where the $\{\hat{P}_i\}$ form the extracted control polygon, the $\{\hat{w}_i\}$ are the associated weights, and the $\{B_{i,k_u}\}$ are the basis functions defined over the knot vector $\{\hat{u}\}$ for a curve of order k_u. The $\gamma_{i^*}(v)$ curve can be formed in similar fashion.

We then calculate the tangent vector by evaluating the derivative of the isocurve at u^*. Using the quotient rule to form the velocity curve for Eq. (2) yields,

$$\gamma'_{j^*}(u) = \left(\frac{\begin{array}{c} \sum_{i=0}^{m} \sum_{j=0}^{m} \hat{P}_i \hat{w}_i \hat{w}_j B_{j,k_u}(u) B'_{i,k_u}(u) \; - \\ \sum_{i=0}^{m} \sum_{j=0}^{m} \hat{P}_i \hat{w}_i \hat{w}_j B_{i,k_u}(u) B'_{j,k_u}(u) \end{array}}{\left(\sum_{j=0}^{m} \hat{w}_j B_{j,k_u}(u)\right)^2} \right).$$

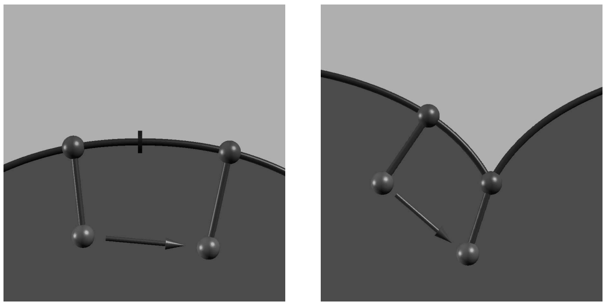
However, since the curve was refined to form an evaluation point, each basis function takes on a value of either zero or one when the velocity curve is evaluated at u^*. The two basis functions that remain active, $B_{i^*,k_u}(u^*)$ and $B_{i^*+1,k_u-1}(u^*)$, result in the simplified expression,

$$\gamma'_{j^*}(u^*) = \frac{(k-1)}{\hat{u}_{i^*+k} - \hat{u}_{i^*+1}} \frac{\hat{w}_{i^*+1}}{\hat{w}_{i^*}} (\hat{P}_{i^*+1} - \hat{P}_{i^*}). \tag{3}$$

Notice that the tangent vector is computed efficiently using only the control polygon, associated weights, and the knot vector. With this and the resulting calculation of Δu and Δv, the new approximation to the local closest point can be evaluated. Since this new point, $S(u^* + \Delta u, v^* + \Delta v)$, is calculated using refinement, the algorithm can continue directly into the next time step.

5.4 Transitioning

Three basic transitions can occur during a trace. Two forms result when the probe's movement causes the local closest point to hit a trim boundary. In the first form of transitioning, the local closest point should transition across the trim and onto an adjacent surface (Fig. 4a). This occurs most often in areas of a model constructed by pasting surfaces together along an adjacent edge. The second possibility is that the closest point should remain on the trim edge. This is the case in concave areas where the probe can trace along the intersection of two surfaces (Fig. 4b). The final form is the special case of transitioning off the model. When the penetration depth becomes negative, tracing ends and tracking resumes, which is effectively the inverse of the contact problem.



(a) (b)

Figure 4. (a) Transitioning across a trimming edge and onto another surface. (b) Transitioning onto the intersection of two surfaces.

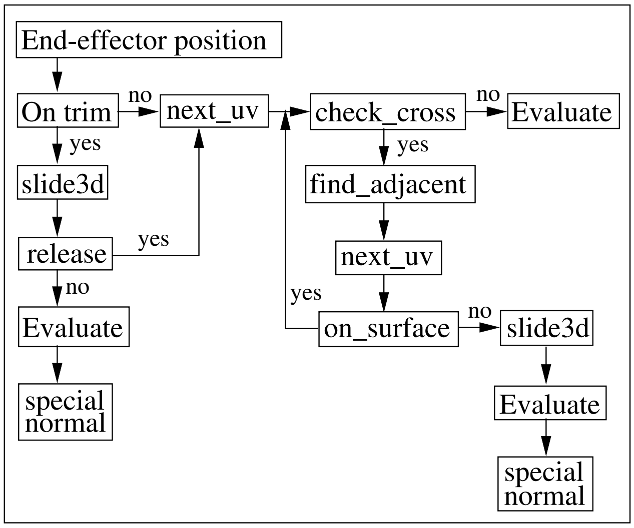


Figure 5. The tracing algorithm follows a different path depending upon when and if the trace results in crossing a trim boundary.

There are four core modules to the tracing algorithm that permit the proper detection and handling of transitions. The `check_cross` module detects when a movement along a surface intersects a trimming loop. Once such an intersection is found the `find_adjacent` module determines the exact corresponding point on the neighboring surface. Tracing along a trim boundary is handled by the `slide3d` module with the `release` module being used to determine when the edge tracing should terminate and normal surface tracing should resume. The path the tracing algorithm takes through these modules is illustrated in Fig. 5.

5.4.1 Trim Intersection.

Discrete movement along the surface correlates to a directed line segment in parametric space. This segment is constructed using the current contact point's parametric coordinates and the next location calculated using direct parametric tracing. If this segment, or *movement vector*, intersects any of the surface's trimming segments then a boundary has been crossed. The location of the intersection is determined by selecting the intersection point closest to the current contact point.

Since the number of trimming segments per surface can be very large, it is not possible to check every segment for intersection. Our solution to this problem is to overlay each surface with a grid. Each cell in the grid contains the trim segments that lie within or intersect it. Ideally, each cell would contain one segment and each segment would be contained in exactly one cell. In practice this is not necessary and would result in heavy preprocessing overhead. We locate the grid so that its boundary coincides with the bounding box of all trim segments. Further, we construct the grid to have four times as many cells as segments with the number of rows and columns being equal. In practice, this is heuristic has proven to be effective.

Each call to `check_cross` results in only checking those segments lying within the cells the movement vector intersects. In addition, we use a grid walking algorithm in order to check these cells in the order the movement vector traverses through them. The intersection checks conclude at the first valid intersection, further cutting down on the number of intersection checks performed.

5.4.2 Adjacency.

In order to smoothly transition from one surface to another it is necessary to calculate an accurate transition point on the neighboring surface. Our system maintains an *edge adjacency table* for each surface. This table allows efficient determination of the adjacent surface as well as the appropriate trimming loop and edge onto which the transition should occur. The segment is found by indexing into the edge to the segment that corresponds to the one intersected. Since adjacent trimming edges run in opposite directions, and the number of segments in these edges is the same for both surfaces, the index of the proper segment can be found directly as $N - i - 1$ where N is the number of segments in the edge and i is the index of the segment intersected. The exact point of transition along this segment, s, is then given by $s(1 - p)$ where p is the percent along the intersected segment that the intersection occurred.

5.4.3 Edge Tracing and Release.

Tracing along a trim edge is closely related to tracing along the surface. The edge tracing algorithm must slide along the edge in Euclidean space to a point locally close to the probes position. Our `slide3d` module does precisely that. The algorithm projects the probe onto the current segment. If this projection remains on the current segment then this projection is our result. If, however, the projection is beyond either endpoint of the segment then we continue to project onto segments along the loop in that direction until a local minimum is found.

Once the local closest point is found the algorithm checks to see if the tracked point should release from the trim. Our algorithm first evaluates one of the surfaces and then uses direct parametric tracing to determine the next location on the surface. If the calculated parametric point is on the surface (i.e. on the correct side of the trimming loop – the right side of the closest segment) then the trace releases from the trim. If the trace does not release onto the first surface then the second is checked. When the trace does not release onto either surface, then a special normal is computed directed from the probe's location to the local closest point. This normal ensures a smooth trace along the edge and also deters further movement into the model.

6 EXTENSIONS

Using the direct haptic rendering algorithm as a black box entity, several worthwhile extensions are possible. We have implemented model manipulation, dimensioned probes, and multi-probe contact. While these are but a few of the potential improvements and extensions, they demonstrate the power and flexibility of the algorithm.

6.1 Models in Motion

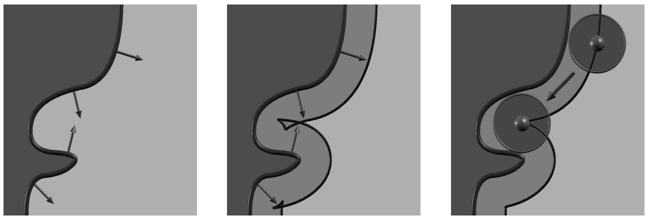
Whether through manipulation, animation, or dynamic properties, mobile models are a fundamental property of virtual environments. The direct haptic rendering algorithm presented in this paper is designed for probe movement with static models, but can be extended so that both probe and models can move. We accomplish this by tracing the original (non-transformed) model with a probe position that has been transformed into model space (Thompson et al., 1997b).

For each model being traced or tracked, we transform the probe through the inverse of the model's transformation matrix. This process transforms the movement of the model into a component of the probe's movement. The resulting closest point and normal are then transformed back from model space to world space. This embedding of the tracing algorithm requires minimal overhead and does not affect the update rate of the haptic process.

6.2 Dimensioned Probe

One of the drawbacks to point probe methods is the dimensionless nature of the probe. If two models are placed directly adjacent to one another, a probe without finite size could still move between the two models without making contact. To eliminate this possibility we compute a model that is projected outward by the radius of the desired probe. A model of this type is often referred to as an offset model (Ho, 1997). Our system uses the offset model in the haptic rendering process while using the original model for the visual display.

Figure 6 illustrates the construction and use of an offset model. The original model in Fig. 6a is offset by the radius of the probe in the direction of the surface normal resulting in the model in Fig. 6b. Isolated regions are trimmed away, producing the offset model in Fig. 6c. Contact with the surface of this offset model represents contact with the original model with a dimensioned probe (Fig. 6c). Notice that any part of the offset model that is trimmed away represents a portion of the original model that could not be contacted with the dimensioned probe. Tracing with a point probe along an edge created by trimming away a region corresponds to tracing multiple contact points of the original model with a dimensioned probe (Fig. 6c).



(a) (b) (c)

Figure 6. (a) Actual model. (b) Initial offset model. (c) Final offset model with possible trace positions.

It is important to note that this process depends on trimming and adjacency information. Further, while this approach uses an auxiliary representation it is not a simplifying "intermediate" representation, since the offset model exactly represents the parts of the original model that can be contacted by the dimensioned probe. Producing the offset model adds significant preprocessing, but it does not affect performance of the tracing algorithm as long as the model geometry does not change during the trace.

6.3 Multi-Probe Contact

Our current implementation of direct haptic rendering of trimmed models runs at over 1000 Hz. However, when using the Sarcos master we notice no improvement when running at any rate over 500 Hz. By running at this lower rate, there is extra time within each cycle of the haptic process. Since the Sarcos device can reflect forces to multiple end-points, we make two calls to the trace algorithm using the location of the finger and thumb as the probe locations. This adds to the overall tracing experience with only a slight data overhead and minimal impact on the tracing algorithms performance. We are currently working on using this additional information to produce stable grasps as well as allow better user control of the models and the environment (Maekawa and Hollerbach, 1998).

7 RESULTS

Our trace algorithm is model-based with a near constant time grid-based transitioning algorithm. This results in equal performance and accuracy for each model, regardless of complexity (Fig. 7).

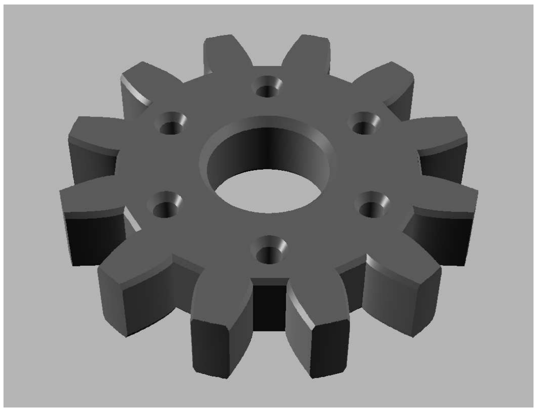


Figure 7. Highly trimmed model of a mechanical gear.

Trimmed NURBS models vary greatly in the number of surfaces and trim segments required to represent them. Table 1 lists a sampling of models against which we tested the system. The *Srfs* column indicates the total number of surfaces for the model. *Segs* indicates the average number of trim segments per surface. Grid statistics are represented in the final three columns. The column labeled *Empty* gives a percentage for the number of cells in a surfaces grid that contain no trim segments. Empty cells translate into essentially zero work for the transitioning algorithm. *Max* gives the maximum number of segments in any one cell. This number represents the worst case for the transitioning algorithm for the given model. Finally, the *Mean* column shows the average number of segments in cells that actually contain segments. This number indicates the amount of work the transitioning algorithm can be expected to perform when near a surface boundary. Note that both the *Max* and the *Mean* columns contain very small numbers in comparison to the *Segs* column. These numbers indicate the drastic reduction in work the transitioning algorithm performs when compared to an algorithm that would check every segment.

The two haptic devices used are very different yet both produced accurate results. The Sarcos Dextrous master is a high inertia device with 10 degrees-of-freedom and a complex dynamics structure. For this device, the haptic process ran on a hybrid PowerPC 604 and Motorola 68040 VME system with a surface stiffness of $6000 N/m$. The Phantom is a low inertia device with 3 degrees of freedom and a rather simple dynamics structure. The haptic process in this instance ran on an SGI Indigo R10000 under IRIX 6.5 with a surface stiffness of $1200 N/m$. Both used a

Model	**Srfs**	**Segs**	**Empty**	**Max**	**Mean**
Goblet	3	254.00	89.92	13	3.38
Brake	28	168.14	72.52	6	1.47
Gear	22	1256.27	92.11	15	4.05
Crank	73	412.00	89.56	36	3.30

Table 1. Statistics on models used in system testing.

nonlinear response model to provide a physically-accurate model of probe-model collision (Marhefka and Orin, 1996). In both cases, the simulation process ran on an Octane with dual R10000 processors and RealityEngine^2 graphics.

Since the quality of the trace is directly related to the calculated closest point and surface normal, we ran simulations to determine the errors for these values. The simulation consisted of a surface trace at a constant penetration depth of $5mm$. The algorithm was able to resolve the closest point to within $0.2mm$ and the error for the surface normal error was under $0.02 degrees$. In practice, the penetration depth averages $3mm$ for the Sarcos master and $1mm$ for the Phantom with low variance. This illustrates that the penetration is not only consistently small, but also consistently near the mean. This combination produces a smooth tracing experience.

8 FUTURE WORK

The goal of our research is to produce a virtual environment that allows intuitive interaction with complex models. To achieve this goal, algorithms to support important capabilities remain under investigation and include:

- An arbitrary probe model would allow the trace to more closely represent the device being used. More complex haptic devices permit tracing with the full hand.
- Collision contact and response for model impacts is a difficult problem, especially when using trimmed NURBS, but a low latency solution needs to be developed for more realistic force response.
- Surface properties such as texture can add to the realism of the tracing experience. Soft and deformable models are a challenge since changes in the geometry can occur during contact.

We are also investigating alternative uses for the tracing algorithm. Among these are methods for modifying arbitrary curves that lie on a surface, sketching on a surface interactively, and finding the silhouette curves of NURBS models.

9 CONCLUSION

We have presented a powerful algorithm that supports the direct haptic rendering of models constructed from trimmed NURBS. This model-based approach permits more complex scenes as well as models to be haptically rendered. Additionally, we have demonstrated the ability of the algorithm to be extended in order to permit model manipulation, tracing by a dimensioned probe, and multi-probe contact. Finally, the distributed system design allows high update rates on both sides of the system, resulting in an interactive visual display coupled with an accurate haptic rendition produced from the actual model.

ACKNOWLEDGMENTS

The authors would like to thank the Biorobotics group for helping operate and setup the robotic devices, their various control systems, and the networking software. Thanks also go to the students and staff of the GDC project, within which this work was developed. Support for this research was provided by NSF Grant MIP-9420352, by DARPA grant F33615-96-C-5621, and by the NSF and DARPA Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-89-20219).

REFERENCES

Adachi, Y., 1993, "Touch And Trace On The Free-Form Surface Of Virtual Object," in *Proc. Virtual Reality Annual Intl. Symp.,* Seattle, WA, pp. 162-168.

Adachi, Y., Kumano, T., and Ogino, K., 1995, "Intermediate Representation For Stiff Virtual Objects," in *Proc. Virtual Reality Annual Intl. Symp.,* Research Triangle Park, NC, pp. 203-210.

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,* Vol. 14, Number 2.

Colgate, J.E., and Brown, J.M., 1994, "Factors Affecting The Z-Width Of A Haptic Display," in *Proc. IEEE 1994 International Conference on Robotics & Automation,* San Diego, CA, pp. 3205-10.

Nelson, D., Johnson, D., Cohen, E., 1999, "Haptic Rendering of Surface-to-Surface Sculpted Model Interaction," in *8th Annual Symp. Haptic Interfaces for Virtual Environment and Teleoperator Systems,* Nashville, TN.

Gibson, J.J., 1966, *The Senses Considered as a Perceptual System,* Boston, MA, Houghton Mifflin Co.

Ho, C., 1997, *Feature-Based Process Planning and Automatic Numerical Control Part Programming,* Ph.D. Thesis, University of Utah, Computer Science Department.

Hollerbach, J.M., Cohen, E.C., Thompson, W.B., and Jacobsen, S.C., 1996, "Rapid Virtual Prototyping Of Mechanical Assemblies," *NSF Design and Manufacturing Grantees Conference,* Albuquerque, NM.

Jacobsen, S.C., Smith, F.M., Iversen, E.K., and Backman, D.K., 1990, "High Performance, High Dexterity, Force Reflective Teleoperator," in *Proc. 38th Conf. Remote Systems Technology,* Washington, D.C., pp. 180-185.

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

Johnson, D.E., and Cohen, E., 1998, "An improved method for haptic tracing of sculptured surfaces," in *7th Annual Symp. Haptic Interfaces for Virtual Environment and Teleoperator Systems,* Anaheim, CA.

Maekawa, H. and Hollerbach, J.M., 1998, "Haptic Display for Object Grasping and Manipulating in Virtual Environment," in *Proc. IEEE Intl. Conf. Robotics & Automation,* Leuven, Belgium, pp. 2566-2573.

Marhefka, D.W., and Orin, D.E., 1996, "Simulation Of Contact Using A Nonlinear Damping Model," in *Proc. International Conference on Robotics and Animation,* Minneapolis, Minnesota, 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," in *Proc. SIGGRAPH 96,* New Orleans, LA, pp. 447-452.

Massie, T.M. and Salisbury, J.K., 1994, "The PHANToM Haptic Interface: A Device for Probing Virtual Objects," in *3rd Annual Symp. Haptic Interfaces for Virtual Environment and Teleoperator Systems,* Chicago, IL, DSC-Vol 1, pp. 295-301.

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

Piegl, L. and Tiller, W., 1995, *The NURBS Book,* Berlin, Springer.

Riesenfeld, R., 1989, "Design Tools For Shaping Spline Models," in *Mathematical Methods in Computer Aided Geometric Design,* (Edited by T. Lyche and L. Schumaker), Academic Press.

Riesenfeld, R., 1993, "Modeling With Nurbs Curves And Surfaces," in *Fundamental Developments of Computer Aided Geometric Design,* L. Piegl (ed.), Academic Press.

Ruspini, D.C., Koloarov, K., and Khatib, O., 1997, "The Haptic Display of Complex Graphical Environments," in *Proc. SIGGRAPH 97,* Los Angeles, CA, pp. 345-351.

Salisbury, K. and Tarr, C., 1997, "Haptic Rendering of Surfaces Defined by Implicit Functions," in *Proc. 6th Annual Symp. Haptic Interfaces for Virtual Environment and Teleoperator Systems,* Dallas, TX, DSC-Vol. 61, pp. 61-67.

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

Stewart, P., Buttolo, P., and Chen, Y., 1997, "CAD Data Representations for Haptic Virtual Prototyping," in *Proc. ASME Design Engineering Technical Conference,* Sacramento, CA.

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

Thompson II, T.V., Nelson, D.D., Cohen, E., and Hollerbach, J.M., 1997, "Maneluverable NURBS Models within a Haptic Virtual Environment," in *Proc. 6th Annual Symp. Haptic Interfaces for Virtual Environment and Teleoperator Systems,* Dallas, TX, DSC-Vol. 61, pp. 37-44.

Zilles, C.B., and Salisbury, J.K., 1995, "A Constraint-Based God-Object Method For Haptic Display," in *Proc. IEE/RSJ International Conference on Intelligent Robots and Systems, Human Robot Interaction, and Cooperative Robots,* Vol 3, pp. 146-151.