A Constraint-based God-object Method For Haptic Display

C. B. Zilles   J. K. Salisbury

Department of Mechanical Engineering
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA

Abstract

Haptic display is the process of applying forces to a human "observer" giving the sensation of touching and interacting with real physical objects. Touch is unique among the senses because it allows simultaneous exploration and manipulation of an environment.

A haptic display system has three main components. The first is the haptic interface, or display device - generally some type of electro-mechanical system able to exert controllable forces on the user with one or more degrees of freedom. The second is the object model - a mathematical representation of the object containing its shape and other properties related to the way it feels. The third component, the haptic rendering algorithm, joins the first two components to compute, in real time, the model-based forces to give the user the sensation of touching the simulated objects.

This paper focuses on a new haptic rendering algorithm for generating convincing interaction forces for objects modeled as rigid polyhedra (Fig. 1). We create a virtual model of the haptic interface, called the god-object, which conforms to the virtual environment. The haptic interface can then be servo-ed to this virtual model. This algorithm is extensible to other functional descriptions and lays the groundwork for displaying not only shape information, but surface properties such as friction and compliance.

1 Introduction

The process of feeling objects through a force-generating interface is familiar in the context of using teleoperator master devices to touch and interact with remotely located objects [7]. Recent interest in enabling interaction with virtual objects [1] has led us to investigate devices and algorithms which permit touch and manipulative interaction – collectively, haptic interactions – with these virtual objects.

The PHANToM haptic interface [4] permits users to feel and control the forces arising from point inter-
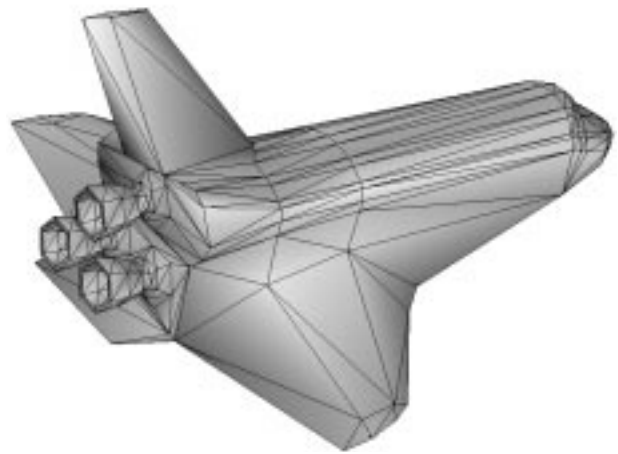

Figure 1: This polygonal model of a space shuttle is made up of 616 polygons. This is an example of the complexity of objects the god-object algorithm can allow the user to touch.

actions with simulated objects. The point interaction paradigm greatly simplifies both device and algorithm development while permitting bandwidth and force fidelity that enable a surprisingly rich range of interactions. It reduces the problem of computing appropriate interaction forces – haptic rendering – to one of tracing the motion of a point among objects and generating the three force components representing the interaction with these objects. In this paper the term *haptic interface point* will be used to describe the endpoint location of the physical haptic interface as sensed by the encoders.

This work was done with PHANToM-style device with a max force of 18N (4 lbf). We have found this to be enough force to make virtual objects feel reasonably solid without saturating the motors. While exploring virtual environments most users tend to use less than 5N (1 lbf) of force.
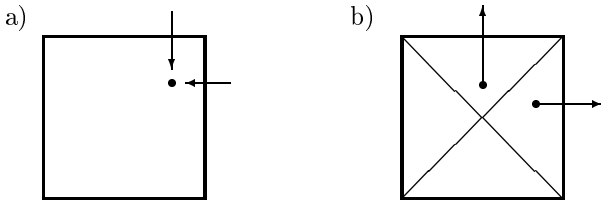
Figure 2: Generation of contact forces using volumes (the small dot represents the haptic interface point). a) Two possible paths to reach the same location in an square are shown; without a history we do not know which path was taken. b) Vector field method (in 2 dimensions): subdivide the square's area and assume that the user entered from the closest edge. The force vectors are normal to the edge and proportional to distance penetrated. This method is easily expanded to 3 dimensions.

Due to the inherent mechanical compliance of haptic interface devices the maximum stiffness of any virtual object is limited. One side-effect of this is that the haptic interface point often penetrates into a simulated object a greater distance than would be possible in real life. In typical demonstrations with our system, the user presses the haptic interface as much as a half an inch into the virtual object when he is stroking the surface. Because humans have poor position sense, this usually goes by unnoticed.

Previous haptic rendering algorithms have tried to determine the feedback force directly from this penetration. These volume methods use a one-to-one mapping of position in space to force. We group these methods together under the label of *vector field methods*.

These vector field methods have a number of drawbacks:

1. It is often unclear which piece of internal volume should be associated with which surface.

2. Force discontinuities can be encountered when traversing volume boundaries.

3. Small and thin objects do not have the internal volume required to generate convincing constraint forces.

This volume penetration can make the logic of generating proper interaction force vectors difficult. When a user is in contact with the center of a large flat surface, it is obvious that the direction of the force should be normal to the plane. The choice of which virtual surface the user should be touching becomes ambiguous as the boundary between surfaces is approached. Multiple paths, meriting different feedback
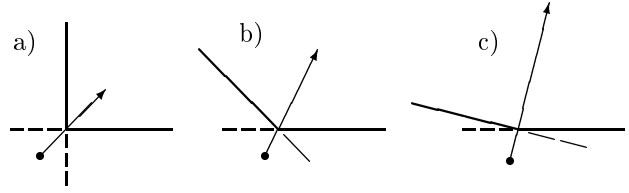

Figure 3: Force summation for multiple objects: a) for perpendicular surfaces, forces sum correctly, but b) as the surface intersection angle becomes more obtuse the force vector becomes too large. Finally c) when the surfaces are almost parallel the force is too large by a factor of 2.

forces, could have been taken to reach the same internal location (Figure 2a). One method [5] suggests subdividing the object volume and associating a sub-volume with each surface (Figure 2b). When inside a sub-volume the force is normal to the associated surface and the magnitude is a function of the distance from the surface, such as with Hooke's law, $F_x = kx$. This method causes a sensation of "sharpness" to be felt at corners due to the sudden force discontinuity from passing from one region into another. This can be useful when truly sharp corners are desired, but confusing when transitions are made accidentally.

This method works for simple geometric shapes because it is reasonably easy to construct these subspaces by hand. In addition, any shape that can be described with an equation can be modeled. For spheres the feedback force direction is that of the vector pointing from the sphere's center to the haptic interfaces point, and the magnitude is a function of the distance the point has penetrated the sphere's surface. The inherent simplicity of these methods has allowed interesting work in dynamic objects and surface effects,[6] but the methods are not flexible enough to allow arbitrary geometries and have some drawbacks.

1.1 Problems with Multiple Objects

Using the vector field approach, it is tempting to construct more complex objects by combining (overlapping) simple objects. In regions where object intersections occur, we might consider computing the net reaction force by vectorially adding contributions from each object's force field in hope that it will generate the correct sensations at corners and edges. This will not always compute the correct force.

When a user is in contact with more than one such object simultaneously, the net surface stiffness can be larger than that of either surface alone. On objects meeting at perpendicular surfaces, the forces can be summed because the distance into the solid is the vector addition of the two orthogonal components (Figure 3). However, when the angle made by the sur-
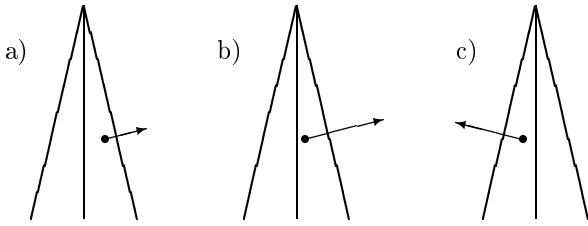
Figure 4: Push through of thin objects. a) user touches surface and feels small force, b) as he pushes harder he penetrates deeper into the object, until c) he passes more than halfway through the object where the force vector changes direction and shoots him out the other side.

faces becomes more obtuse, the resulting reaction force magnitude (and corresponding apparent stiffness) approaches twice that of either surface alone. Surfaces intersecting at acute angles are less stiff than either surface alone. The system is no longer robust; these stiffness discontinuities could exceed the maximum stable stiffness.

Generating believable forces requires directly determining the distance the haptic interface point has penetrated an object and the surface(s) it has passed through to arrive at its current position. The constraint-based method described in Section 2 computes this distance directly.

1.2   Problems with Thin Objects

Vector field methods also break down when thin objects are to be rendered. Because of the limited servo and mechanical stiffnesses, the haptic interface point must travel somewhat into the object before enough force can be applied to make the object feel "solid". When this distance becomes greater than the thickness of an object, the vector field model produces unrealistic sensations. As shown in Figure 4, when the haptic interface point reaches halfway through the object, it is pulled through the rest of the way. The algorithm has assumed that the user entered through the other side. This effect takes place at any convex corner.

One possible solution is to keep a history of contact occurrences so we know the surfaces our haptic interface point has passed through. With a history of old locations it would be clear which surface is meant to be touched and, therefore, which force should be displayed; unfortunately, this method rapidly becomes numerically cumbersome. It is important to have a compact representation in both use of memory and processing time. We describe below the use of a "god-object" as a compact representation of history.

2   The God-object Algorithm

We will define and explain the reasoning behind the god-object representation and how it can be utilized in rendering polyhedral representations of objects, in Section 2.1. The term god-object has been previously used [2] in a similar spirit to describe a virtual object controlled by a human user in physical simulations.

Using the history (the god-object location calculated in the previous servo cycle) and the current haptic interface point, a set of surfaces currently impeding motion can be found. A discussion of constraints is given in Section 2.2.

Lagrange multipliers are used to find the new location of the god-object during contact with a virtual object. The god-object's new location is chosen to be the point which locally minimizes the distance between the god-object and the haptic interface point, subject to the constraint that the god-object is *on* a particular surface. The mathematics of this method are explained in Section 2.3

2.1   God-objects

As we saw in the previous section a number of problems arise from the penetration of the haptic interface point into virtual objects. We know we cannot stop the haptic interface point from penetrating the virtual objects, but we are free to define additional variables to represent the *virtual location* of the haptic interface. This location is what we will call the *god-object*.

We have complete control over the god-object; we can prevent it from penetrating any of the virtual objects and force it to follow the laws of physics in the virtual environment. The god-object is placed where the haptic interface point would be if the haptic interface and object were infinitely stiff. Because the god-object remains on the surface of objects, the direction of the force should never be ambiguous. This allows a more realistic generation of the forces arising from touching an object. In particular, this method is suitable for thin objects and arbitrarily shaped polyhedra.

In free space, the haptic interface point and the god-object are collocated, but as the haptic interface moves into an object the god-object remains on the surface. The god-object location is computed to be a point on the currently contacted surface such that its distance from the haptic interface point is minimized. This assumes the god-object moves across the surface without being impeded by friction; inclusion of friction is a simple extension.[8]

By storing additional state variables for the position of the god-object (one variable for each degree of freedom of the apparatus) we can keep the useful

history of the object's motion in a compact manner. In our work with a three-degree-of-freedom haptic interface, the god-object is a point needing only three coordinates to fix its location.

Once the god-object location is determined, simple impedance control techniques can be used to calculate a force to be displayed. A stiffness and damping can be applied between the haptic interface point and the god-object, representing local material properties. The stiffness and damping can vary between objects and across an object as long as they do not exceed the maximum displayable values of the device as limited by servo stiffness and stability. In addition, non-linear stiffnesses could be used to give surfaces more interesting sensations, such as the click of a button.

2.2 Constraints

Although we are interested in simulating volumes, we interact with those volumes on their surfaces. In general, it is more convenient to represent objects by their surfaces. To simplify the mathematics of the problem, only planar surfaces are used.

In this work we have found that a good first-cut haptic representation can be derived from the same polyhedral geometry used to represent objects for visual rendering. Straightforward lists of vertices, edges and facet orientation, as are found in standard polyhedral representations, are sufficient to permit use of the god-object algorithm. This is particularly convenient in that it enables haptic rendering of a large body of existing visually render-able objects.

A mesh of triangular elements is used because it is the most fundamental, and assures that all of the nodes are coplanar. Graphic models do not require the exactness required by haptic models, so it is not uncommon to find objects with four-noded surfaces where the nodes are not coplanar. The problems caused by such surfaces can be avoided by using a triangular mesh. In addition, since a plane is completely determined by three points, we can move nodes *on the fly* and still have geometrically acceptable surfaces. Moving nodes due to applied forces is one way to implement deformable surfaces.

Using a polygonal representation of objects makes collision detection simple. For an infinite surface (a planar constraint), we will denote the surface as *active* if the old god-object is located a positive (in the direction of the surface normal) distance from the surface[1],
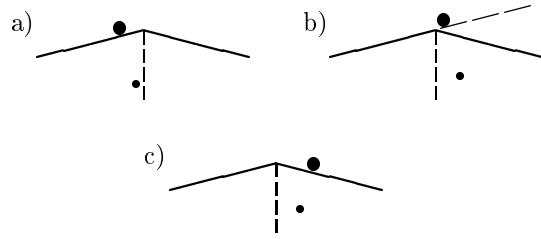

Figure 5: Motion between convex surfaces takes place in two steps (the small dot represents the location of the haptic interface, and the larger dot represents the god-object): a) approaching the edge, b) the left surfaces is still active so the new god-object location is places in the plane of the left surface, but not in its boundaries, c) now that the god-object is free of the left surface it can fall to the right surface. This distortion of shape is generally imperceptible.

and the haptic interface point has a negative distance to the surface (i.e. on the other side). This creates surfaces as one-way constraints to penetration[2].

When the surfaces are not of infinite extent, then for a surface to be active, we also require that the god-object contact take place *within the boundaries* of the surface. A line can be traced from the old god-object to the new haptic interface point. If this line passes through the facet (*within all of the edges*) then that facet should be considered as active.

When touching convex portions of objects, only one surface should be active at a time. The transition of the god-object between two surfaces sharing a convex edge requires two steps (Figure 5). While the first surface is active, the contact point must stay on the plane of the surface, but not necessarily within the boundaries; the first step places the god object over the second surface, but still in the plane of the first surface. In the next servo loop the first surface will no longer be active and the god-object can then fall to the second surface. The times and distances involved are small enough to cause only an imperceptible and transient distortion of shape.

When probing a concavity, multiple surfaces can be active simultaneously. When touching the concave

---

[1] Due to the round-off error present in any digital system the newly computed god-object position might be infinitesimally below the virtual surface. If no precautions are taken the god-object will cross (and no longer be restrained by) the virtual surface. To prohibit such behavior, we add a small constant to the calculated distance from the god-object to the virtual surface.

Conceptually, this is equivalent to using another plane just below the virtual surface for computation of god-object distances.

[2] This method treats surfaces as single-sided surfaces respecting the fact that objects are formed from a closed set of surfaces. There is no physical reason to touch both the inside and outside of an object simultaneously. In some respects it is actually beneficial because a simulation can begin with the haptic interface in any configuration; if the haptic interface starts inside a virtual object the user can simply move out of the object as it were free space. In vector field methods a "workaround" has to be introduced for this contingency.
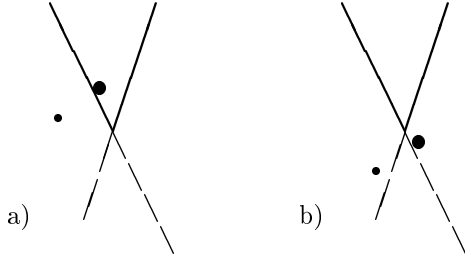
Figure 6: For an acute concave intersection of surfaces the god-object will be able to cross one of the surfaces unless special precautions are taken. The large dot represents the position of the god-object and the small dot represents the haptic interface. a) If the user is pressing into one surface and sliding down, b) the god-object will cross to the negative side of the surface before the haptic interface will and the constraint will not be activated. To prevent this we use an iterative routine.

intersection of two planes, both constraints are active, and the god-object's motion should be restricted to a line (the intersection of both planes). When in contact with the intersection of three surfaces, all three will be active and the net constraint becomes a point (the intersection of all three planes). At the intersection of more than three surfaces, only three will be active at any one time and the user will still be constrained to a point. Once we identify one active constraint, we can temporarily limit our search to neighboring surfaces.

Additional care must be taken when surfaces intersect at an acute angle (as viewed from the "outside" of the surfaces) to form a concavity. If the user presses into one such surface and slides along it, it is possible for the god-object to cross the other constraint surface before the haptic interface point does (Figure 6). Once the god object crosses a wall, it will be free of that constraint; therefore, we must not allow the crossing.

One solution is to iterate the whole process. The first iteration will find a set of active constraints and calculate a new god-object location. Using the "new" god-object location as the haptic interface point, the constraints of neighboring surfaces are checked again to see if any additional surfaces should be active. If additional constraints are found, then another "new" god-object location is computed. This iteration continues until no new constraints are found. This iteration process requires very little time, since the number of possible constraints in the neighborhood of a contact is very small. The maximum number of iterations is equal to the maximum number of possible simultaneous constraints.

2.3 God-object Location Computation

When a set of active constraints has been found, Lagrange multipliers can be used to determine the location of the new god-object. Equation (1) gives the energy in a virtual spring of unity stiffness, where x, y, and z are the coordinates of the god-object and x_p, y_p, and z_p are the coordinates of the haptic interface point. Constraints are added as planes because they are of first order (at most) in x, y, and z in the form shown in equation (2). The general case for our 3-degree-of-freedom haptic interface involves 3 constraints; for less constrained instances, zeros are used to replace unused constraints giving a lower order system.

$$Q = \frac{1}{2}(x - x_p)^2 + \frac{1}{2}(y - y_p)^2 + \frac{1}{2}(z - z_p)^2. \quad (1)$$

$$A_n x + B_n y + C_n z - D_n = 0 \quad (2)$$

The new location of the god-object is found by minimizing L in equation (3) below by setting all six partial derivatives of L to 0. Because the constraints are of first order and Q is of second order, the differentiation is very simple. In fact, the differentiation can be done symbolically and the actual coefficient values can be just substituted at runtime. In the general case we have six variables (x, y, z, l_1, l_2, l_3), which will give us six partial derivatives organized into a set of simultaneous equations:

$$
\begin{aligned}
L \;=\; & \tfrac{1}{2}(x - x_p)^2 + \tfrac{1}{2}(y - y_p)^2 + \tfrac{1}{2}(z - z_p)^2 \\
& + l_1(A_1 x + B_1 y + C_1 z - D_1) \\
& + l_2(A_2 x + B_2 y + C_2 z - D_2) \\
& + l_3(A_3 x + B_3 y + C_3 z - D_3)
\end{aligned} \quad (3)
$$

$$
\begin{bmatrix}
1 & 0 & 0 & A_1 & A_2 & A_3 \\
0 & 1 & 0 & B_1 & B_2 & B_3 \\
0 & 0 & 1 & C_1 & C_2 & C_3 \\
A_1 & B_1 & C_1 & 0 & 0 & 0 \\
A_2 & B_2 & C_2 & 0 & 0 & 0 \\
A_3 & B_3 & C_3 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
x \\ y \\ z \\ l_1 \\ l_2 \\ l_3
\end{bmatrix}
=
\begin{bmatrix}
x_p \\ y_p \\ z_p \\ D_1 \\ D_2 \\ D_3
\end{bmatrix}
\quad (4)
$$

The matrix, equation (4), has a number of useful properties. It is symmetric, the upper left hand corner (3x3) is always the identity matrix, and the lower left hand corner is always a null matrix, it should never require row swapping, and should always be invertible.[3] Because of these properties it requires only 65 multiplicative operations (multiplies and divides) to solve

---

[3] It is important to make sure that none of the surfaces are parallel because this will cause the matrix to be singular.

for x, y, and z. In the 5x5 case (when there are only 2 constraints), it requires only 33 multiplicative operations, and in the 4x4 case (a single constraint), it takes only 12. When no constraints are active, the god-object is located at the position of the haptic interface. Also because of its symmetry, and the identity and null portions, we only need variables for 15 of the 36 matrix locations and 6 for the right side vector.

2.4 Performance

The above described algorithm has been implemented in C++ to run on a 66MHz Pentium, interfaced to a PHANToM haptic interface.

For objects with about 600 triangular facets the simulation runs at about 1 kHz. Much of the time is spent running the simple collision detection algorithm. With an improved collision detection algorithm [3] significant speedups can be expected. Due to the local nature of the god-object scheme, only the collision detection part of the problem scales with the size of the model.

3 Conclusion

The constraint based god-object method allows a user to intuitively control a point probing a virtual object and prevents the point from penetrating these objects. Because the god-object remains on the surface of the virtual objects, it is easy to compute realistic force vectors to apply to the user. In addition, the method allows use of the extensive libraries of polygon-meshed objects already in existence. Furthermore, the god-object implementation provides, with no additional computation, coordinates of the contact point suitable for visual display of haptic interactions.

Our current research focuses on adding a number of important haptic effects to the above algorithm. [6] While the current algorithm faithfully reproduces the sharpness occuring at the discontinuity between adjacent surfaces, we have found that it is possible to interpolate surface normals between surfaces to smooth out these transitions in much the same way as is currently the practice with visual shading algorithms (e.g. Phong shading). We have also demonstrated that commonly encountered haptic sensations such as friction, texture, and surface impedance can be convincingly displayed by careful modulation of the force vector applied to the user. While such effects so far have been limited to simple demonstrations with object shapes not requiring the generality of the god-object algorithm, we expect they will become part of a more general haptic rendering engine currently under development.[8]

References

[1] N. Durlach et al, *Virtual Reality: Scientific and Technological Challenges, Report produced for the National Research Council*, National Academy of Sciences, Washington D.C. December 1994.

[2] P. Dworkin and D. Zeltzer, "A New Model for Efficient Dynamic Simulation", Proceedings Fourth Eurographics Workshop on Animation and Simulation, pp.135-147, 1993.

[3] M. C. Lin, D. Manocha, and J. Canny, "Fast Collision Detection between Geometric Models," Tech Report TR93-004, Department of Computer Science, University of North Carolina, 1993.

[4] T. Massie, "Design of a Force Reflecting Fingertip Stimulator", *SB thesis*, MIT EECS Department, May, 1993.

[5] T. Massie and K. Salisbury, "The PHANToM Haptic Interface: A Device for Probing Virtual Objects," *Proceedings of the ASME Winter Annual Meeting*, Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Chicago, IL, November 1994.

[6] K. Salisbury, D. Brock, T. Massie, N. Swarup, C. Zilles "Haptic Rendering: Programming Touch Interaction with Virtual Objects," to appear in the *Proceedings of the ACM 1995 Symposium on Interactive 3D Graphics*, Monterey CA, April 1995.

[7] T. Sheridan, *Telerobotics, Automation, and Supervisory Control*, MIT Press, Cambridge, MA, 1992.

[8] C. Zilles, "Haptic Rendering using the Toolhandle Haptic Interface", *MS-SB thesis*, MIT Mechanical Engineering Department, May, 1995.