Copyright 1996, Published in the Proceedings of the First PHANToM Users Group Workshop, M.I.T. Artificial Intelligence Laboratory Technical report AITR-1596, 1996.

Robust Haptic Display of Graphical Environments

Diego C. Ruspini[1], Krasimir Kolarov[2] and Oussama Khatib[1]

Robotics Laboratory[1]
Stanford University, Stanford, CA 94305

Interval Research Corporation[2]
1801 Page Mill Road, Palo Alto, CA 94304

Introduction

We are developing a new haptic interface library "HL." With a Application Programmers Interface (API) almost identical to that of "GL", a graphics hardware interface library used on all Silicon Graphics workstations, haptic environments can quickly and efficiently be incorporated into graphics applications. The library uses a multi-level control system to effectively simulate contact with virtual environments. Work has also begun to help guarantee that the behavior of the haptic device is stable even if the simulated model is too complex to be simulated accurately.

In our implementation contact, friction and other forces are not applied directly to the haptic device, but to a virtual "proxy" or representative object in the virtual world. This object, similar to the god-object proposed by Zilles and Salisbury [Zilles94], has a known mass, velocity and acceleration and can therefore be accurately simulated as it is effected by forces in the virtual environment. The haptic device and a virtual spring/damper are then used to keep the error between the virtual and real objects configuration to a minimum.

To allow complex models to be simulated, fast collision detection between the proxy and the virtual environment is essential. In our implementation a hierarchy of bounding spheres is automatically constructed when objects are introduced [Quinlan94]. This hierarchy can be quickly traversed to prune objects that are too far from the proxy to effect its movement. Only nearby obstacles need to be checked fully to see if they lie in the path of the proxy. The use of hierarchical bounding volumes reduces the computation costs enough to allow complex environments to be simulated.

Many graphic interfaces, including "GL," allow modelers to specify surface normals on the vertices of polygonal surfaces. This information is used to alter the lighting model on the surface to give it the appearance of being smooth. The normal information is used by our system to create a similar haptic effect. The constrained motion of the proxy is computed using the local surface orientation inferred by the supplied normals. Once the constrained motion is found, it is projected back onto the actual object surface. The proxy position is then advanced along this projected motion direction. This gives the user the appearance that they are on a smooth surface while actually moving along the underlying polygonal model. To simulate smooth surfaces, a force shading algorithm was proposed in [Morgenbesser96].

By using a virtual proxy the haptic servo controller task is reduced to minimizing the error between the configuration of the proxy and position of the haptic device. Reducing position error of a mechanical system is problem that has been dealt with extensively in the robotics literature [Craig89]. In our current implementation a strait forward operational space proportional derivative (PD) controller is used to guarantee stability even in the presence of a large number of objects. The low level control loop can be separated from the contact/proxy update loop. By running the control loop at a high fixed clock rate stability is more easily ensured and the fidelity of the haptic display can be made to degrade gracefully, as the complexity of the environment is increased.

2. The Client Application

The "HL" Library allows users to define objects as a collection of primitive objects -- points, line segments, or polygons. Transformations are provided to allow objects and primitives to be translated or rotated freely. Surface normal and texture coordinates can be associated with polygons vertices to allow smooth or textured surfaces to be defined. Any number of objects can be created, and objects can call other objects allowing the creation of complex object hierarchies. All primitives specified between the time an object is opened and closed belong to the new object. The new object does not become visible to the haptic display until it is closed. This condition ensures that a given object is not made touchable until it is completely specified. A special object (Object 0) is always displayed. This object, possibly empty, forms the root of the object hierarchy. This slight difference from the "GL" specification allows the object hierarchy to be continuously defined, unlike in "GL" where time is divided into discrete frames. This continuity in the definition of the virtual environment is important since the haptic server may be running at a servo rate many times that of the application program. On the client side almost all functions are small wrapper procedures that pack the functions arguments into a packet to be transmitted to the haptic server.

3. Model Construction

Copyright 1996, Published in the Proceedings of the First PHANToM Users Group Workshop, M.I.T. Artificial Intelligence Laboratory Technical report AITR-1596, 1996.

Once the modeling commands are received from the client, they must be stored in a form suitable for haptic rendering. Vertices are transformed into the local object frames and meshes and sequences of line segments are broken into a set of independent convex bodies. Intersections tests between the convex primitives and the path of the virtual proxy are performed by the Gilbert distance algorithm described in [Gilbert88]. This algorithm computes the distance between two convex polyhedrons in $O(n+m)$ time, where n and m are the number of vertices in each polyhedra. Each polyhedra is specified as the convex hull of a set of vertices. No additional information about the edges or the facets of the convex hull, or the order of the vertices is required. This algorithm returns the nearest point to each polyhedra as the affine combination of the vertices of the polyhedra. The affine weights can be used to interpolate surface normals and texture coordinates from the vertices of the underlying polygon. While this algorithm is more general then what is required by our current system, in the future we hope to extend our system to allow the proxy to be represented a more complex object.

Because each object is normally constructed from a large number of primitives, a naive test of checking each primitive with the path of the proxy would be prohibitively expensive. In general the proxy will be in contact with at most a small fraction of the underlying primitives. To make use of the spatial coherence inherent in the object a hierarchical bounding representation for the object is constructed. The bounding representation is based on spheres similar to the one described in [Quinlan94]. This hierarchy of bounding spheres is constructed by first covering each polygon with small spheres in a manner similar to scan conversion in computer graphics. These spheres are the leaves of an approximately balanced binary tree. Each node of this tree represents a single sphere that completely contains all the leaves of its descendants.

After covering the object a divide and conquer strategy is used to build the interior nodes of the tree. This algorithm works in a manner similar to quick-sort. First an axis aligned bounding box that contains all the leaf spheres is found. The leaf spheres are then divided along the plane through the mid-point of the longest axes of the bounding box. Each of the resulting two subsets should be compact and contain approximately an equal number of leaf spheres. The bounding tree is constructed by recursively invoking the algorithm on each subset and then creating a new node with the two subtrees as children.

Two heuristics are used to compute the bounding sphere of a given node. The first heuristic finds the smallest bounding sphere that contains the spheres of its two children. The second method directly examines the leaf spheres. The center is taken as the mid-point of bounding box already computed earlier. The radius is taken to be just large enough to contain all the descendant leaf nodes. Note that a given node is not required to fully contain all of its descendant nodes only their leaf nodes. The method that generates the sphere with the smallest radius is used for the given node. The first heuristic tends to work better near the leaves of the tree, while the second method produces better results closer to the root. This algorithm has an expected $O(nlogn)$ execution time, where n is the number of leaf spheres.

4. Updating Proxy Position

At each time step the virtual proxy moves in the direction that reduces the error between it and the actual finger position, subject to the constraints imposed by the objects in the environment. In our current implementation the proxy is modeled as a sphere with a radius specified by the user. Because of small numerical errors polygons that are intended to share a common edge may in fact contain gaps. The proxy should therefore be at least large enough to avoid slipping through these holes in the underlying model, otherwise no restriction is placed on the size of the proxy. Most often a user would like to make the proxy large enough to be easily visible on a graphics display.

To determine if a collision occurs between the proxy and an object in the environment, the volume sweep by the virtual proxy, as it moves during this time period, is checked to see if it penetrates any primitive in the environment. In our system the time step is always small enough so that the path of the proxy can be approximated by a straight line segment. Because the proxy is modeled as a sphere the comparison is reduced to determining if a line segment comes within one proxy radius of a given polygon. A sphere containing this line segment is checked with the bounding sphere hierarchy of each object. If the two spheres do not touch, then the underlying primitives cannot interfere with the path of the proxy, and no further testing is required. If the spheres penetrate, a collision is still possible and the child nodes of the object are checked recursively to see if they also touch the bounding sphere of the path. If a leaf node is reached the proxy path and primitive are checked using the Gilbert distance algorithm. Because a primitive may have more then one leaf sphere, and the path bounding sphere may intersect more then one of these leaves, a comparison cache is kept to quickly determine if a distance test has already been performed at this time step, thus avoiding extraneous calls to the distance algorithm.

When a collision between a primitive and the proxy path is found, a plane normal to the obstacle in the configuration space of the proxy is calculated at the point where the proxy will first make contact with the obstacle. Once all contacts have been found the proxy position can be moved until it makes contact with the closest contact planes. Planes that fall below this new point cannot at least locally affect the motion of the proxy and can therefore be pruned. If the proxy reaches the users position no further

Copyright 1996, Published in the Proceedings of the First PHANToM Users Group Workshop, M.I.T. Artificial Intelligence Laboratory Technical report AITR-1596, 1996.

movement is required. Otherwise a new constrained motion direction is needed. In [Zilles94] Zilles and Salisbury propose using Lagrange multipliers to find the constrained motion of the "god-point." The paper describes a solution where the allowable region of motion is restricted to the intersection of all the contact planes, when in fact at most three of the constraint planes can be active at one time.

Determining the contact plane can be solved with standard quadratic programming techniques like those described in [Gill86], however this problem has many simplifications that make a less complex and faster solution possible. In our implementation this problem is solved by taking its dual. The solution to this problem is independent of translation or scaling, thus the proxy can be shifted to the origin and the distance from the proxy point to the desired goal position can be scaled to one. All contact planes, since they are in contact with the proxy, now go through the origin. In this dual each contact plane is represented by a point, defined by the negative components of the contact plane normal. The negated normals represent the outward normals of the local-free space region. The active constraint planes are found by finding the point nearest to the goal position on the convex hull of the contact plane points and the origin. An example of this duality is illustrated in Figure 1 below.

This EPS image does not contain a screen preview.

It will print correctly to a PostScript printer.

File Name : duality.ps

Title : /robo_net/robo2/u1/ruspini/duality.ps

CreationDate : 10:38 PM September 23, 1996

Pages : 1

Figure 1: Duality of Constrained Motion Solution

This problem is now in a form that can be solved using the same distance algorithm as was used for the primitive contact test. This algorithm returns the nearest point as the affine combination of points on the convex hull. Each point with a positive contribution to the solution indicates that the corresponding contact plane constrains the motion of the proxy. By using the dual this problem can be solved in O(n) time, where n is the number of contact planes and without the addition of any large software package. Once the (at most) three active constraint planes are found the desired motion of the proxy can be quickly found and the iteration can continue.

5. Force Shading

Most graphic interfaces allow modelers to specify surface normals on the vertices of polygonal surfaces. This information is used to alter the lighting model on the surface to give it the appearance of being smooth. To create a similar haptic effect, when contact between the proxy and a polygon (containing user specified normals) is detected, a new normal at the surface is calculated. This is done by interpolating the specified vertex normals using the weights (returned by the distance algorithm) that specify the contact point. This interpolation is very similar to the interpolation done for Phong shading in computer graphics. The computed normal is used to specify a new contact plane going through the contact point. This plane is used instead of the actual contact plane to compute the motion of the proxy as described above. Once the constrained motion is found, this motion is projected back onto the actual contact surface. This process is illustrated in Figure 2.

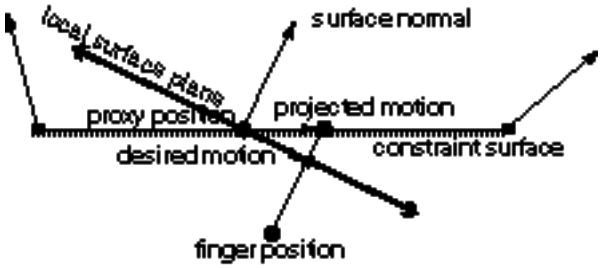

Figure 2: Proxy Update with Force Shading

The difference between the simulated smooth surface and a normal surface is illustrated in figures 3 and 4. In the example the difference between the actual user position and the position of the virtual proxy are shown as the users finger follows a circular path around a ten sided polygonal approximation of a circular object. For compactness only the first quadrant of the circle is shown. Because the constraint surface is curved, the differences between the position of the proxy and the finger are best illustrated in polar coordinates. The angular displacement corresponds roughly to the amount to which the users finger is pulled tangentially as the user moves around the circle. The radial displacement indicates roughly the amount of force the user feels pulling the finger toward the surface of the object.

As is seen in Figure 3 a strong discontinuity occurs when the proxy finally reaches the edge of the object. In Figure 4 surface normals have been specified on the vertices as shown. The result movement of the proxy shows that no tangential force is felt by the user as the finger is moved around the object. This is what would be expected if the user where moving around a perfectly circular object. In the radial plot it can be seen that the proxy still follows the underlying surface of the object. However the motion of the proxy has no discontinuities and therefore the surface appears smooth to the user.

Copyright 1996, Published in the Proceedings of the First PHANToM Users Group Workshop, M.I.T. Artificial Intelligence Laboratory Technical report AITR-1596, 1996.

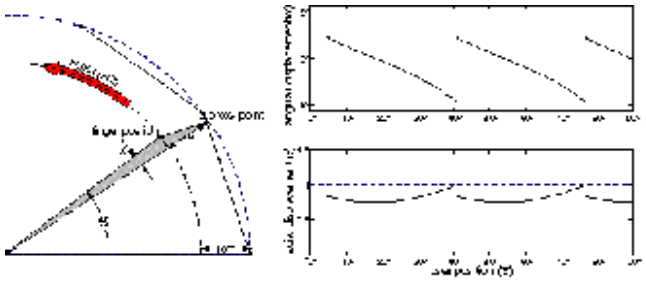

Figure 3: Virtual Proxy Path

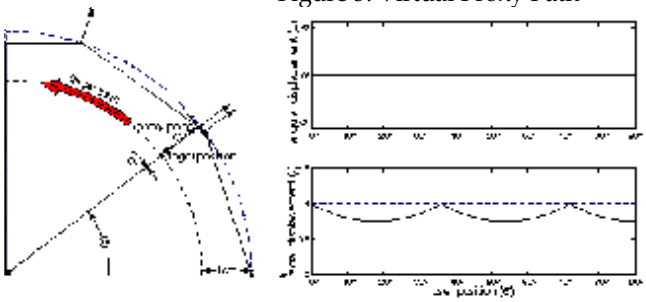

Figure 4: Proxy Path with Force Shading

It should be noted that this technique may increase the distance between the user position and the proxy. This makes ensuring stability much more difficult. It is possible to specify normals and polygons that make the resultant motion of the proxy unstable. However for normals typically seen in practice proxy movement appears to be well behaved.

6. Haptic Control

By using a virtual proxy the haptic servo controller task is reduced to minimizing the error between the configuration of the proxy and position of the haptic device. Reducing position error of a mechanical system is problem that has been dealt with extensively in the robotics literature. In our current implementation a simple operational space proportional derivative (PD) controller is used to guarantee stability even in the presence of a large number of objects. The low level control loop can be separated from the contact/proxy update loop. By running the control loop at a high fixed clock rate stability is more easily ensured and the fidelity of the haptic display can be made to degrade gracefully, as the complexity of the environment is increased.

7. Results

Our haptic library has been successfully used on models containing more the 2000 polygonal primitives. In our tests the client computer was an SGI Indigo2 High Impact running IRIX 6.2. The haptic server was a 90Mhz Pentium running Linux 2.0.2, connected to a PHANToM haptic device. Communication was through a standard ethernet TCP-IP connection. The haptic server gave stable results for that many polygons even with position gains over 1600 Newton/meter, and no artificial damping. The cycle time of the proxy update loop shows only an approximately O(logn) growth with the number of polygons. It is hoped that with a faster CPU, even more complex models can be simulated.

Future Work

At present our current library does not allow objects to be moved or manipulated. This is a large drawback since one of the main goals of our project is to allow people to better interact with the virtual environment. Work to allow the proxy to interact with the objects in the environment has already begun.

Bibliography

[Gilbert88] E. G. Gilbert, D. W. Johnson and S. S. Keerthi, "A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space," IEEE J. of Robotics and Automation, Vol.4, No. 2, April 1988.

[Gill86] P. Gill, S. Hammarling, W. Murray, M. Saunders and M. Wright, "User's Guide to LLSOL," Stanford University Technical Report SOL 86-1, (January 1986).

[Massie94] Massie, T.M., Salisbury, J.K., "The PHANToM Haptic Interface: A Device for Probing Virtual Objects." ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems 1994, In Dynamic Systems and Control 1994 (Chicago, Illinois, Nov. 6-11), vol. 1, pp.295-301.

[Morgenbesser96] Morgenbesser, H.B., Srinivasan, M.A., "Force Shading for Haptic Shape Perception." ASME Winter Annual Meeting 1996.

[Quinlan94] Quinlan, S., "Efficient Distance Computation between Non-Convex Objects," Int. Conference on Robotics and Automation, (April 1994).

[Zilles94] Zilles, C. B., Salisbury, J. K., "A Constraint-based God-object Method for Haptic Display." ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems 1994, In Dynamic Systems and Control 1994 (Chicago, Illinois, Nov. 6-11), vol. 1, pp.146-150.