Six-DOF Haptic Rendering II

CPSC 599.86 / 601.86
Sonny Chan - University of Calgary
Outline

- Generic model for proxy-based rendering
- Study of three significant 6-DOF haptic rendering algorithms
  - McNeely, Puterbaugh, & Troy 1999
  - Otaduy & Lin 2005
  - Ortega, Redon, & Coquillart 2007
Controller ORTEGA ET AL.: A SIX DEGREE-OF-FREEDOM GOD-OBJECT METHOD FOR HAPTIC DISPLAY OF RIGID BODIES WITH SURFACE...

edge, and face can potentially be felt. while providing the user with precise haptic display, where each vertex, with and slide on the environment obstacles without penetrating them, computation method allows the manipulated object to come in contact haptic display of contacting rigid bodies (here, two Stanford bunnies described in this paper allows us to provide a user with high-quality and details several future research directions. Finally, Section 8 concludes limitations of our approach. Section 7 demonstrates our approach producing haptic effects for surface perception such as force applied to the user. Section 6 discusses methods for constraint-based quasi-static approach to computing the motion of the god-object to ensure realistic haptic interaction with rigid bodies. Section 5 presents our novel method for three degree-of-freedom haptic rendering of generic polygonal objects. They introduced the representation of the haptic device that is constrained to the surface of the obstacles. In their three degree-of-freedom approach, the location of the god-object minimizes at each time step the distance to the haptic object. The device; the difference between the two positions provides that is constrained to remain on the surface of the obstacles (configuration...}

Recent work on stable six degree-of-freedom interactions by these two configurations. Our method extends the classical three degree-of-freedom constraint-based method by Zilles and Salisbury [2] by proposed a general constraint-based method for a six objects or objects parts [6], thereby degrading the perceptible force artifacts (see discussion in Section 7). Some authors have proposed six degree-of-freedom approaches (i.e., interpenetrations, forces felt at a distance, or artificial friction and sticking). These methods, like most six degree-of-freedom haptic display methods [13], [14], [15], [16], [17], [18], do not attempt to prevent the interpenetration between the virtual objects, which might lead to missing some collisions through the use of an implicit integration method. The paper is organized as follows: Section 2 provides a brief review of our approach. Section 3 describes the virtual proxy approach to a three degree-of-freedom interaction with objects defined by implicit methods to smooth the object surface and add friction. Ruspini et al. [6] extend this approach by replacing the god-object by a small sphere and propose the force direction. Niemayer and Mitra [7] propose dynamic proxies to better extend the virtual proxy approach to a three degree-of-freedom approach, the location of the god-object is constrained to the surface of the obstacles. In their use of an implicit integration method. Haptic display of virtual objects has been an active area of research over the last decade. In 1995, Zilles and Salisbury proposed what appears to be the first constraint-based haptic rendering method that does not suffer from the visual or haptic artifacts of previous constraint-based haptic rendering method that does not artifacts created by a virtual coupling can be reduced. The methods, like most six degree-of-freedom haptic display algorithms. McNeely et al. [10] propose a voxel sampling method. Johnson et al. [11] use local minimum distances to compute the force applied to the user based on the discrepancy between two rigid reference frames: one attached to the haptic device, and one attached to the virtual object. We typically place the origin at the center of gravity of the virtual object, although any point can be chosen. Only the god-object is displayed (and not the actual virtual objects, which might lead to missing some collisions through the use of an implicit integration method. The paper is organized as follows: Section 2 provides a brief review of our approach. Section 3 describes the virtual proxy approach to a three degree-of-freedom interaction with objects defined by implicit methods to smooth the object surface and add friction. Ruspini et al. [6] extend this approach by replacing the god-object by a small sphere and propose the force direction. Niemayer and Mitra [7] propose dynamic proxies to better extend the virtual proxy approach to a three degree-of-freedom approach, the location of the god-object is constrained to the surface of the obstacles. In their use of an implicit integration method. Haptic display of virtual objects has been an active area of research over the last decade. In 1995, Zilles and Salisbury proposed what appears to be the first constraint-based haptic rendering method that does not suffer from the visual or haptic artifacts of previous constraint-based haptic rendering method that does not artifacts created by a virtual coupling can be reduced. The methods, like most six degree-of-freedom haptic display algorithms. McNeely et al. [10] propose a voxel sampling method. Johnson et al. [11] use local minimum distances to compute the force applied to the user based on the discrepancy between two rigid reference frames: one attached to the haptic device, and one attached to the virtual object. We typically place the origin at the center of gravity of the virtual object, although any point can be chosen. Only the god-object is displayed (and not the actual virtual objects, which might lead to missing some collisions through the use of an implicit integration method.
Virtual Coupling

Dynamic model based on virtual coupling.

\[ \mathbf{F}_c = k_T \mathbf{x} + b_T \mathbf{v} \]

\[ \boldsymbol{\tau}_c = k_R \mathbf{\theta} + b_R \mathbf{\omega} \]
Goal is to compute position, orientation of the proxy, given

- Applied force, torque from virtual coupling
- Contact forces or constraints
Dynamic Proxy Simulation

\[ F = F_c + \sum_i F_i \]

\[ = ma \]

\[ \tau = \tau_c + \sum_i M_i \]

\[ = I_{CM} \alpha + \omega \times I_{CM} \omega \]
Time Integration

- Explicit Euler finite difference equation:
  \[ y_{n+1} = y_n + \Delta t \dot{y}_n \]

- with the state variable

\[
y(t) = \begin{pmatrix} x \\ \theta \\ P = m v \\ L = I \omega \end{pmatrix} \quad \dot{y}(t) = \begin{pmatrix} \dot{x} \\ \dot{\theta} \\ \dot{P} \\ \dot{L} \end{pmatrix} = \begin{pmatrix} \frac{1}{m} P \\ \omega \\ F \\ \tau \end{pmatrix}
\]
Comments on Virtual Coupling

- The spring-damper coupling filters high frequency force variations (or discontinuities) applied to the virtual tool
  - Can be a good or a bad thing...
- A stiffer coupling spring allows the operator to feel more of the contact forces
- However, stiff coupling springs can lead to instabilities in free space (why?)
Limitations of Time Integration

What happens with a harmonic oscillator?
Implicit Time Integration

- Implicit Euler finite difference equation:
  \[ y_{n+1} = y_n + \Delta t \dot{y}_{n+1} \]

- Using first order Taylor approximation:
  \[ y_{n+1} = y_n + \Delta t \left[ \dot{y}_n + \frac{\partial \dot{y}}{\partial y} (y_{n+1} - y_n) \right] \]
  \[ \left( I - \Delta t \frac{\partial \dot{y}}{\partial y} \right) (y_{n+1} - y_n) = \Delta t \dot{y}_n \]
Summary

- Implicit Euler integration is much more stable than explicit integration
- Undershoots rather than overshoots
- Requires computing the derivatives (Jacobian) of the force vector with respect to state variables
- Allows use of stronger penalty forces, stiffer virtual coupling
Collision Detection

- Mesh-mesh collision detection was the thoughest in the book!

- Dynamic proxy solver also requires penetration depth

- Poses the greatest challenge to 6-DOF haptic rendering...
Collision Detection Approaches

- Recall 1000 Hz update rate requirement for haptic rendering
  - How can we possibly get it fast enough?

- Many approaches, but we will examine two:
  - Simplify or modify geometric representation
  - Run collision detection at a lower rate if needed
Voxmap PointShell™

[From W.A. McNeely et al., Proc. SIGGRAPH, 1999.]
Voxelized Geometry

- Point-voxel collision tests are fast
- Idea: Voxelize all the geometry

Polygonal model, “Voxmap”, and “PointShell” representations of a teapot
Computing the Voxmap

<table>
<thead>
<tr>
<th>Offset Layers</th>
<th>Exact Surface</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Force Layer</th>
<th>Surface Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>BAD</td>
</tr>
</tbody>
</table>

0 = free space
1 = interior
2 = surface
3 = proximity
Computing the PointShell

- Approximate with centers of surface voxels
- Add inward-pointing surface normals
Collision Response

- Virtual tool is dynamically simulated, so we can apply forces to it
- Use tangent-plane force model and Hooke’s Law
Collision Response

- Net force on virtual tool is sum of penalty forces from point-voxel intersections

- **Problem**: What happens with multiple, simultaneous contacts?

- **Solution**:

\[
F_{net} = \begin{cases} 
F_{total}, & N < 10 \\
\frac{F_{total}}{\frac{1}{10}N}, & N \geq 10 
\end{cases}
\]
Collision Response

- Another **problem**: Can a point-voxel intersection occur on an interior voxel?

- **Solution**: Apply a “braking viscosity” force at the proximity voxels.

\[
F = \begin{cases} 
-bv(-n \cdot v), & n \cdot v < 0 \\
0, & n \cdot v \geq 0 
\end{cases}
\]

- Large point velocities are still a problem...
Summary

- This rendering method can provide a constant 1000 Hz update rate that includes collision detection (on a 350 MHz PC!)
- Resolution is limited by voxel size, and finer voxel grids use cubically more memory
- Many problems with ad-hoc solutions...
- Still one of the first highly successful 6-DoF rendering techniques
Stable & Responsive Manipulation

Sensation-Preserving Simplification

- Finding all contact points between detailed polygonal models can be really expensive!
- Take advantage of perceptive limitations

Collision Detection Strategy

- Create multi-resolution hierarchies of the meshes (levels of detail)
- Accelerate collision detection with BVH
- Only refine search where details are perceptible!
Constructing the Hierarchy

- Perform full convex decomposition on original mesh

- Then start merging pieces in priority of highest resolution (most detail)
  - Perform filtered edge collapse decimation to simplify components while preserving convexity

- Mark as level of detail whenever number of components is halved
Levels of Detail

LOD hierarchy doubles as bounding volume hierarchy!
Collision Detection

- Traverse BVH as usual for collision detection, except...
- Only recurse when the higher resolution is deemed perceptible
- Otherwise, use approximate geometry at the current LOD
Variable Rate Collision Detection

- Still cannot guarantee speed!
- As low as 100 Hz with 40k triangles

- Haptic thread can render forces at 1000 Hz while contact thread runs at a variable rate
Collision Response

- Remember problem with multiple contacts?
- K-means clustering is used to group contacts into representative points
- Each cluster described by point and normal
- Viscoelastic penalty-based force applied to the virtual tool for each contact:

\[
F_p = -kN(x + Rr - p_0) - kdN - bN(v + \omega \times r)
\]

\[
T_p = (Rr) \times F_p
\]
Summary

- Adaptive simplification = fast collision detection between complex models
- Fidelity of haptic perception is preserved
- Variable rate collision detection allows high force and haptic update rate
- Contact clustering mitigates force discontinuities and escalating stiffness for multi-point contact
Dynamic Proxy Limitations

- Did we solve the interpenetration problem?
  - Nonpenetration enforced by high contact stiffness, can cause instability

- Are there other limitations?
allowing the user to feel the bumps and holes defined by this function.

This PC communicates with a cluster of PCs only dedicated processor Xeon PC, to which the haptic device is connected. The entire algorithm is executed on a 3.2 GHz dual-

user to interact intuitively on a large two-screen display six degree-of-freedom force-feedback device, allows a Haptic Workbench in which the SPIDAR-G, a tension-based

The validation of our approach is performed on a Stringed 7R update rate of the simulation loop.

However, similar to the Minsky approach, if the speed of the god-object is too high, or the update rate of the

be mixed with the force shading effect described above.

perturbation vector at each contact point.

modify the force vector direction is defined by averaging multiple contact points, the perturbation vector used to

bumps and holes along this axis (cf. Fig. 9). In the case of function along one axis could be sufficient for providing

computed by the six degree-of-freedom constraint-based

high-frequency textures.

and a force feedback. This produces a convincing effect of

device location and the map to provide a surface property

proposed by Blinn. The approach combines the haptic

synthesize high-frequency textures for a haptic device. Only

[32]. Minsky [33] was the first to introduce a system to

textured surfaces are in three degrees of freedom [30], [31],

here, most of the existing approaches proposed to explore

force. The perturbed force

entry of the sine function to find a value for perturbing the direction of the

A similar effect can be produced by perturbing the force

Bump and Hole Texture.

Fig. 9.

This method provides high-frequency textures and can

A similar effect can be produced by perturbing the force

Fig. 10. (From M. Ortega et al., IEEE Trans. Visualization and Computer Graphics 13(3), 2005.)

6-DOF God-Object
Constraint-Based Proxy Solver

- Direct analogue of 3-DOF god-object
- Uses contact positions and normals only – presumes objects do not interpenetrate
- Computes a trajectory that does not violate contact constraints
Contact Constraints

How do we use these to determine the motion of the proxy?

\[ a_{CM} \cdot \hat{n}_k + \alpha \cdot (r_k \times \hat{n}_k) \geq 0 \]
Gauss’ Principle of Least Constraint

- Gauss defined a kinetic distance quantity as

\[
\mathcal{G}(a) = \frac{1}{2} (a - a^u)^T M (a - a^u)
\]

\[
= \frac{1}{2} \|a - a^u\|_M^2
\]

- Then the motion of the constrained body is one that minimizes the kinetic distance

\[
a^c = \arg \min_a \mathcal{G}(a)
\]
Quasi-Static Proxy Update

- Write the generalized accelerations as

\[ \mathbf{a} = (a_{CM}, \alpha)^T \]

- Obtain unconstrained acceleration from virtual coupling spring (proxy displacement)

\[ \mathbf{a}^u = \frac{1}{2} (\mathbf{x}_h - \mathbf{x}_s) \]
Solve the quadratic programming problem

\[
\begin{align*}
\text{minimize} \quad & G(a) = \frac{1}{2} (a - a^u)^T M (a - a^u) \\
\text{subject to} \quad & a_{CM} \cdot \hat{n}_k + \alpha \cdot (r_k \times \hat{n}_k) \geq 0
\end{align*}
\]

Then update the proxy with the constrained motion (possibly with additional collision query)

\[
x'_s = x_s + \frac{1}{2} a^c
\]
other words, the constrained acceleration following function [23]:

\[ a \approx \text{euclidean projection of the unconstrained acceleration} \]

... microseconds constraint-based force applied to the user within a few based coupling loop, and allows us to compute the suppresses the need for collision detection in the constraint-based coupling loop. Essentially, the constraint-based coupling device and the contact information sent by the god-object applied to the user based on the configuration of the haptic constrained motion of the god-object.

Necessary and sufficient information to compute the

\[ \text{constraint-based force to be applied to the user is then} \]

by solving Gauss' projection problem. The con-strained acceleration of the god-object is com-

\[ \text{puted from the unconstrained acceleration} \]

... of the god-object minimizes the generalized acceleration of the god-object: \( \frac{1}{J} \phi a_c \).

Note that the matrices... solved using Wilhelmsen's projection algorithm onto the set of possible accelerations. This projection

\[ a_c \]

where

\[ a \]

\( \frac{1}{C^2} \)

... of the god-object is com-

\[ \text{the matrices} \]

... of constraints to the constraint-based coupling loop. This flag, written to the

\( F \)

... configuration reachable by the god-object, while haptic

\[ \text{configurations 3 and 4, which correspond to accelerations} \]

... that do not generate... contact point (the diagonal line). The possible accelerations

\[ \text{main computation involved is the determination of the} \]

... in the case of a god-object in contact with an obstacle. For clarity, only two... degrees of freedom are allowed: a vertical translation and a rotation whose axis is orthogonal to the plane of the figure.

Fig. 4 demonstrates this algorithm in the case of a god-object with four degrees of freedom: \( \text{translational and rotational} \)

\[ a^u \]

... force to be applied to the user and smooth the constraint-based force applied to the user and

\[ \text{efficiently (see Section 7).} \]

The configuration of the haptic device changes, and the only the configuration of the haptic device... do not have to be updated either.

\[ \text{Fig. 4a shows the god-object contacting the obstacle (in blue)} \]

... (rotation whose axis is orthogonal to the plane of the figure). The possible accelerations... Fig. 4b shows the corresponding two-dimen-

\[ \text{smooth the constraint-based force applied to the user and} \]

... a vertical translation and a rotation whose axis is orthogonal to the plane of the figure.
Continuous Collision Detection

- Constraint-based proxy solver requires non-interpenetrating contacts

- Continuous collision detection is one method to find contacts and normals while enforcing non-interpenetration
Recall 3-DOF God-Object

- The segment-triangle intersection test is a form of continuous collision detection
- The god-object is infinitely small, so it will always miss polygonal geometry unless CCD is used!
Non-Point Proxies

How do we generalize to a polyhedral avatar?

[From S. Redon et al., Transactions of the ASME 5, 2005.]
Arbitrary In-Between Motions

- We only know the position of the avatar at discrete time steps
- We may assume an arbitrary object motion subject to:
  - Interpolation
  - Continuity
  - Rigidity

\[ P_t \rightarrow P(t) \rightarrow P_{t+1} \]
Interpolating Motion

- Describe a continuous equation for rigid-body motion between the two known positions:
  \[ T(t) = c^0 + t(c^1 - c^0) \]
  \[ R(t) = \cos(\omega t)(I - uu^T)R^0 + \sin(\omega t)u^*R^0 + uu^TR^0 \]

- where \( \omega \) is rotation angle and \( u \) is the rotation axis between configurations
Testing for Intersection

- Edges intersect at $t$ if

$$\vec{a}(t) \vec{c}(t) \cdot \left( (\vec{a}(t) \vec{b}(t)) \times (\vec{c}(t) \vec{d}(t)) \right) = 0$$

- Vertex/face intersect at time $t$ if

$$\vec{a}(t) \vec{b}(t) \cdot \left( (\vec{b}(t) \vec{c}(t)) \times (\vec{b}(t) \vec{d}(t)) \right) = 0$$

- How do we find $t$?
Interval Arithmetic

\[ I = [a, b] = \{ x \in \mathbb{R}, a \leq x \leq b \} \]

\[ [a, b] + [c, d] = [a + c, b + d] \]
\[ [a, b] - [c, d] = [a - d, b - c] \]
\[ [a, b] \times [c, d] = [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)] \]
\[ 1/[a, b] = [1/b, 1/a] \quad \text{if } a > 0 \text{ or } b < 0 \]
\[ [a, b] / [c, d] = [a, b] \times (1/[c, d]) \quad \text{if } c > 0 \text{ or } d < 0 \]
\[ [a, b] \leq [c, d] \quad \text{if } b \leq c \]
Solving for Intersection

- To Solve:
  \[
  \overline{a(t)c(t)} \cdot \big( \overline{a(t)b(t)} \times \overline{c(t)d(t)} \big) = 0, \quad \overline{a(t)b(t)} \cdot \big( \overline{b(t)c(t)} \times \overline{b(t)d(t)} \big) = 0
  \]

- Use interval arithmetic, evaluate using the interval \( t = [0, 1] \)

- If zero is in the result interval, halve and repeat:
  
  \[
  t = [0, 1] \quad t = [0, \frac{1}{2}] \quad t = [\frac{1}{2}, 1] \\
  t = [0, \frac{1}{4}] \quad t = [\frac{1}{4}, \frac{1}{2}] \quad t = [\frac{1}{2}, \frac{3}{4}] \quad t = [\frac{3}{4}, 1]
  \]
Bounding Volumes

- Continuous collision detection also works with bounding volume intersection tests
- For example, the sphere test becomes

\[ \| c_1(t) - c_2(t) \| \leq r_1 + r_2 \]

\[ (c_2(t) - c_1(t))^2 \leq (r_1 + r_2)^2 \]

- Conservative test:
  - There may be an intersection if the lower bound on the left is less than the right side
CCD Summary

- Finds the first contact between the avatar and the scene along a motion path

- Not quite “continuous”, but computes time of contact to a precision

- Can combine with structures like BVHs
Collision Detection Performance

- Fast, but not haptic rates for large meshes
- Execution time varies:
  - 70 Hz for 27k triangles
- Again, use multiple threads at different rates...
4.1 Overview

Our algorithm is divided in three asynchronous loops: (1) the haptic display loop which computes the force applied to the user, (2) the constraint-based coupling loop which updates the current configuration of the haptic device, and (3) the 6DOF simulation loop which computes the acceleration of the god-object.

Implementation Diagram

- **Haptics Loop** (1 ms)
- **Constraint-based Coupling** (~ µs)
- **6DOF God-Object Simulation** (~ ms)

The matrices $G$, $C_1$, and $C_2$ are computed based on the current contact information. The Nonpenetration constraint is expressed as $\mathbf{n}^T \mathbf{a} = 0$, where $\mathbf{n}$ is the normal vector to the environment obstacle, and $\mathbf{a}$ is the constrained acceleration of the god-object. The acceleration of the god-object is computed from the position and orientation of the haptic device and the environment obstacles.

The nonpenetration constraint is directed toward the exterior of the environment obstacle, and the number of contact points can be easily determined from the contact positions and normals. Continuous collision detection algorithms provide the necessary data to ensure that the god-object does not penetrate the environment obstacles.

The matrices $M$, $C_1$, and $C_2$ are updated based on the current contact information. The matrices $G$, $C_1$, and $C_2$ are concatenated to form a single constraint matrix $\mathbf{A}$, where $\mathbf{A} = \mathbf{G} + \mathbf{C}_1 + \mathbf{C}_2$. The unconstrained acceleration is directed toward the exterior of the environment obstacle, and the number of contact points (see details below) can be easily determined from the contact positions and normals.

The nonpenetration constraint is expressed as $\mathbf{n}^T \mathbf{a} = 0$, where $\mathbf{n}$ is the normal vector to the environment obstacle, and $\mathbf{a}$ is the constrained acceleration of the god-object. The acceleration of the god-object is computed from the position and orientation of the haptic device and the environment obstacles.
Summary

- Advantages:
  - Continuous collision detection ensures no object penetration
  - No forces are felt in free space

- Disadvantages?
Proxy-Based Rendering

Device Controller

Virtual Coupling

Proxy Solver

Collision Detector

1. position + orientation
2. force + torque
3. position + orientation
4. contact points + normals or forces

Device

Virtual

Proxy

Collision

Controller

Coupling

Solver

Detector

Diagram with arrows between the components indicating the flow of data: position, orientation, force, torque, and contact points.
# Proxy Rendering Taxonomy

<table>
<thead>
<tr>
<th></th>
<th>Soft Constraints</th>
<th>Hard Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Massless Proxy</strong></td>
<td>Quasi-Static Equilibrium</td>
<td>Distance Minimization</td>
</tr>
<tr>
<td><strong>Proxy with Mass</strong></td>
<td>Penalty-Based Dynamics</td>
<td>Constrained Dynamics</td>
</tr>
</tbody>
</table>

- **Proxy Rendering Taxonomy**
- **Soft Constraints**
  - Quasi-Static Equilibrium
- **Hard Constraints**
  - Distance Minimization
  - Constrained Dynamics