Six-DOF Haptic Rendering II
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
The paper is organized as follows: Section 2 provides a summary of related work. Section 3 gives an overview of constraint-based quasi-static approach to computing the motion of the god-object to ensure realistic haptic interaction with rigid bodies. Section 4 describes how we compute the generic polygonal objects. They introduced the idealized representation of the position of the haptic device that is constrained to the surface of the obstacles. In their approach, an idealized representation of the position of the haptic device is constrained to remain on the surface of the obstacles (configuration $x_s$), the god-object minimizes at each time step the distance to the haptic device penetrates the environment obstacles (configuration $x_{h}$). We propose new algorithms to compute the motion of the god-object to ensure realistic haptic interaction with objects defined by implicit representations [8], [9].

Recent work on stable six degree-of-freedom interactions by Otaduy and Lin [20], however, has shown that the force applied to the user seems to be the first six degree-of-freedom constraint-based haptic rendering method that does not suffer from the visual or haptic artifacts of previous methods, like most six degree-of-freedom haptic display algorithms. McNeely et al. [10] propose a general constraint-based method for a six degree-of-freedom interaction with rigid bodies. However, 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 and can lead to the well-known pop-through effect or artificial friction and sticking).

Although the haptic device can traverse thin edges, 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 with rigid bodies (here, two Stanford bunnies described in this paper allows us to provide a user with high-quality shading and textures. Section 7 demonstrates our approach producing haptic effects for surface perception such as force applied to the user. Section 6 discusses methods for extending 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. Several authors have proposed to replace the god-object by a small sphere and propose the force direction. Ruspini et al. [6] extend this approach by device; the difference between the two positions provides the user with a high-quality haptic display of contacting rigid bodies. We also discuss the benefits and limitations of our approach. Finally, Section 8 concludes with a discussion of future research directions.
Virtual Coupling

\[ F_c = k_T x + b_T v \]
\[ \tau_c = k_R \theta + b_R \omega \]
Goal is to compute position, orientation of the proxy, given

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

Proxy Solver

force + torque

position + orientation

position + orientation

contact points + normals or forces
Dynamic Proxy Simulation

\[ \mathbf{F} = \mathbf{F}_c + \sum_i \mathbf{F}_i \]
\[ = m \mathbf{a} \]
\[ \boldsymbol{\tau} = \boldsymbol{\tau}_c + \sum_i \mathbf{M}_i \]
\[ = \mathbf{I}_{CM} \alpha + \omega \times \mathbf{I}_{CM} \omega \]
Time Integration

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

- with the state variable

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

- position + orientation
- contact points + normals or forces
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>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

0 = free space
1 = interior
2 = surface
3 = proximity

Exact Surface

Force Layer

Surface Layer

OK

BAD
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

\[ F = K_{ff} \cdot d \]
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:

\[
F_p = -kN(x + Rr - p_0) - kd_n - bN(v + \omega \times r) \\
T_p = (Rr) \times F_p
\]
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?
6-DOF God-Object

Contraint-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?

\[ \mathbf{a}_{CM} \cdot \hat{n}_k + \alpha \cdot (\mathbf{r}_k \times \hat{n}_k) \geq 0 \]
Gauss’ Principle of Least Constraint

- Gauss defined a kinetic distance quantity as

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

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

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

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

- Write the generalized accelerations as
  \[ \mathbf{a} = (\mathbf{a}_{CM}, \alpha)^T \]

- Obtain unconstrained acceleration from virtual coupling spring (proxy displacement)
  \[ \mathbf{a}^u = \frac{1}{2} (\mathbf{x}_h - \mathbf{x}_s) \]
Optimization Problem

- Solve the quadratic programming problem

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

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

\[ x'_s = x_s + \frac{1}{2} a^c \]

CS277 - Experimental Haptics, Stanford University, Spring 2014
Constrained Motion

Our method uses Gauss' least constraints principle to compute the constrained motion of the god-object. The configuration of the god-object is fixed, but this might lead to constraint forces that are not orthogonal to the velocity of motion.

For clarity, only two degrees of freedom are allowed: a vertical translation and a rotation. Fig. 4b shows the corresponding two-dimensional unconstrained accelerations and forces.

When a new set of constraints is available, some of the matrices and the god-object configuration do not have to be updated either.

Note that, because the configuration of the haptic device changes, and the nonpenetration constraint resulting from the constraints to the constraint-based coupling loop. This might create a large constraint force if the user has not been able to smooth the constraint-based force applied to the user and suppresses the need for collision detection in the constraint-based quasi-static computations as in the god-object simulation loop, but this loop performs the same constant force transmission.

Data retrieval: The configuration reachable by the god-object, while haptic configurations 1 and 2 result in a force and a torque which attempt to bring the haptic device back to a position where it can slide freely along the wall. Haptic configurations 3 and 4, which correspond to accelerations 4 and 5, do not have to be updated either.

Fig. 4 demonstrates this algorithm in the case of a god-object in contact with an obstacle. For clarity, only two nonpenetration constraints might not be satisfied by the current configuration of the haptic device (see Fig. 6a). This might create a large constraint force if the user has not been able to smooth the constraint-based force applied to the user.
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

\[ \mathbf{P}_t \]

\[ \mathbf{P}(t) = ? \]

\[ \mathbf{P}_{t+1} \]
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

\[ x(t) = P(t)x \]
Testing for Intersection

- Edges intersect at $t$ if
  
  \[
  \mathbf{a}(t) \mathbf{c}(t) \cdot \left( \mathbf{a}(t) \mathbf{b}(t) \times \mathbf{c}(t) \mathbf{d}(t) \right) = 0
  \]

- Vertex/face intersect at time $t$ if
  
  \[
  \mathbf{a}(t) \mathbf{b}(t) \cdot \left( \mathbf{b}(t) \mathbf{c}(t) \times \mathbf{b}(t) \mathbf{d}(t) \right) = 0
  \]

- 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:

\[
\mathbf{a}(t) \mathbf{c}(t) \cdot (\mathbf{a}(t) \mathbf{b}(t) \times \mathbf{c}(t) \mathbf{d}(t)) = 0, \quad \mathbf{a}(t) \mathbf{b}(t) \cdot (\mathbf{b}(t) \mathbf{c}(t) \times \mathbf{b}(t) \mathbf{d}(t)) = 0
\]

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

- If zero is in the result interval, halve and repeat:

  - \( t = [0, 1] \)
  - \( t = [0, \frac{1}{2}] \)
  - \( t = [\frac{1}{2}, 1] \)
  - \( t = [0, \frac{1}{4}] \)
  - \( t = [\frac{1}{4}, \frac{1}{2}] \)
  - \( t = [\frac{1}{2}, \frac{3}{4}] \)
  - \( t = [\frac{3}{4}, 1] \)
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...
Implementation Diagram

Haptics Loop

Constraint-based Coupling

6dof God-Object Simulation

Shared Data

(1 ms)  (≈ μs)  (≈ ms)
Summary

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

- Disadvantages?
Controller

ORTEGO 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, computation method allows the manipulated object to come in contact containing about 27,000 triangles each). Our constraint-based force 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 motion of the god-object to ensure realistic haptic interaction with objects defined by implicit and the force applied to the user based on the discrepancy between two rigid reference frames: one virtual proxy can traverse thin represent a generic polygonal objects. They introduced the method for three degree-of-freedom haptic rendering of [2] proposed what appears to be the first constraint-based research over the last decade. In 1995, Zilles and Salisbury Haptic display of virtual objects has been an active area of recent work on stable six degree-of-freedom interactions by 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, artifacts of previous approaches (i.e., interpenetrations, forces felt at a distance, artifacts created by a virtual coupling can be reduced or artificial friction and sticking). These methods, like most six degree-of-freedom haptic display algorithms. McNeely et al. [10] propose a general constraint-based method by Zilles and Salisbury [2] by Our method extends the classical three degree-of-freedom constraint-based quasi-static approach to computing the

Fig. 1. Six degree-of-freedom god-object.

Fig. 2. Proxy-Based Rendering

proxy: position + orientation

force + torque

position + orientation

device: force + torque

virtual coupling: position + orientation

proxy solver: contact points + normals or forces

collision detector: position + orientation

Device Controller

Virtual Coupling

Proxy Solver

Collision Detector

Device Controller

Virtual Coupling

Proxy Solver

Collision Detector

CS277 - Experimental Haptics, Stanford University, Spring 2014
# Proxy Rendering Taxonomy

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

CS277 - Experimental Haptics, Stanford University, Spring 2014