Exploiting Perception in High-Fidelity Virtual Environments

Introduction to Haptic Rendering

Ming C. Lin
Department of Computer Science
University of Maryland at College Park
http://www.cs.umd.edu/~lin
lin@umd.edu

What Is Haptic Rendering?

Simulation
Haptic Device
Virtual Reality

Master-Slave Systems
Human-in-the-loop

Robot
Haptic Device
Human

Force Feedback
Tactile Feedback

Pictures: http://haptic.mech.nwu.edu/intro/gallery/

Inter-disciplinary Research

Computer Science
Electrical Engineering
Mechanical Engineering

Haptic Rendering
Control and actuators
Mechanical design

Computation of the forces output by the device

Control of Haptic Devices

• Impedance Devices
  6-DOF Phantom
• Admittance devices
  COBOTs

Collaborative Haptic Design Review

Engine Close-Up

Boeing VPS System
Other Examples

- A Haptic Hybrid Controller for Virtual Prototyping of Vehicle Mechanisms (Ford, BMW, etc)
- 3-DOF Cobot for Engineering Design (Northwestern University and Ford Automobile)

Immersion Medical Simulators

- Endoscopy simulator - Bronchoscopy and upper and lower gastrointestinal procedures on a single platform
- Endovascular simulator - Percutaneous coronary and peripheral interventions and cardiac rhythm management
- Hysteroscopy simulator - Skills assessment and myomectomy
- Laparoscopy simulator - Skills, cholecystectomy, sterilization, ectopic pregnancy, and myomectomy suturing
- Vascular access simulator - Adult, geriatric, and pediatric IV; PICC; phlebotomy; and skills assessment

Virtual Endoscopic Surgery Training

VEST System One (VSO One) Technology

- 3 haptic (force-feedback) devices as mock-up endoscopic instruments
- 1 virtual endoscopic camera
- three new Basic Task Training (BTT) exercises - Find tubes/touch points/follow path

Laparoscopic Surgery

- MIT Touch Lab

Molecular Dynamics

- VMD: Visual Molecular Dynamics

Haptic Vector Field

- Lawrence, Lee, Pau, Roman, Novoselov
  - University of Colorado at Boulder
- 5 D.O.F. in
- 5 D.O.F. out
Basic Pipeline

User

motion

forces

Haptic Device

motion

command

position

Grasped Object

Scene

Haptic Rendering Loop

User

forces

Haptic Device

motion

command

position

Grasped Object

Scene

Problem of Haptic Rendering

1. The user becomes part of the simulation loop.
2. 1KHz is necessary so that the whole system doesn’t suffer from disturbing oscillations.
   - Think of the analogy with numerical integration of a system with spring, mass and damper, where the frequency of the haptic loop sets the integration step.
3. The Phantom haptic devices run their control loop at 1KHz.
4. Consequence: we are very limited on the amount of computation that we can do.

Key Challenges

- Collision Detection
  - Choice of representation and algorithm
- Interaction Paradigm
  - Penalty forces vs. constraint-based optimization
  - Virtual coupling vs. direct rendering
  - Newtonian dynamics / Quasi-static approximation
  - Single user vs. collaboration
Additional Issues

- Decouple haptic and simulation loops?
  - Use intermediate representations?
- Force type and quality
  - How hard does hard contact feel?
  - How free does free-space feel?
    - Repulsive forces?
    - Force artifacts / stability considerations

3DOF Haptics: Introduction

- Output: 3D force -> 3DOF haptics
- Limited to applications where point-object interaction is enough.
  - Haptic visualization of data
  - Painting and sculpting
  - Some medical applications

3DOF Haptics: Basic approach

- Check if point penetrates an object.
- Find closest point on the surface.
- Penalty-based force.
\[ F = k \cdot x \]

3DOF Haptics: The problems

- Force discontinuities when crossing boundaries of internal Voronoi cells.
- Unexpected force discontinuities (both in magnitude and direction) are very disturbing!

3DOF Haptics: God-object

- Use the position of the haptic interface point (HIP) and a set of local constraint surfaces to compute the position of god-object (GO).
- Constraint surfaces defined using heuristics.
- Compute GO as the point that minimizes the distance from HIP to the constraint surfaces. Lagrange multipliers.
3DOF Haptics: God-object

- Constraint surfaces:
  - Surfaces impeding motion
  - GO is outside (orientation test) and in the extension of the surface.
  - The HIP is inside the surface.

3DOF Haptics: Virtual proxy

- Ruspini et al., SIGGRAPH 1997.
- Based on god-object.
- Virtual proxy is a small sphere, instead of a point. Use configuration-space obstacles (C-obstacles), from robotics.
- More formal definition of constraint planes.
- Implementation of additional features, based on relocation of the virtual proxy.

3DOF Haptics: Virtual proxy

- C-obstacles: for a spherical object, is reduced to computing offset surfaces at a distance equal to the radius of the sphere.
- Check the HIP against the offset surface.
- This is done to avoid problems with small gaps in the mesh.

3DOF Haptics: Virtual proxy

- Finding the virtual proxy is based on an iterative search.
- Basically, find subgoals based on the same distance minimization as for the god-object.
- At each subgoal, all the planes that go through that point are potential constraints. The minimum set of active constraints is selected.
- If the subgoal is in free space, set as new subgoal the HIP. The path might intersect the C-obstacles. Add the first plane intersected as a constraint and the intersection point as the current subgoal.
- The process ends when the virtual proxy becomes stable.

3DOF Haptics: Virtual proxy

- HIP(t)
  - Perform collision detection between the path of the HIP and the C-obstacles
  - HIP(t+\Delta t)

3DOF Haptics: Virtual proxy

- HIP(t)
  - Set the subgoal and the constraint plane(s)
  - HIP(t+\Delta t)
3DOF Haptics: Virtual proxy

Find a new subgoal using the active planes and the minimization based on Lagrange multipliers.

Since the subgoal is in free space, drop the constraints, set the HIP as the new subgoal and perform collision detection between the path and the C-obstacles.

The path to the new subgoal intersects another plane, so this is added to the set of constraints.

Quadratic programming approach:
- The constraint planes define an open convex region (bounded by the plane at infinity).
- The function to minimize is the distance from the haptic device (HIP) to the new subgoal ($VP_{i+1}$):
  \[ C = \left\| VP_{i+1} - HIP(t + \Delta t) \right\| \]
- The translation from the current location to the new subgoal cannot intersect the constraint planes. Define linear constraints based on the normals of the planes.
  \[ N_j \cdot (VP_{i+1} - VP) \geq 0 \]
3DOF Haptics: H-Collide

- Collision Detection for 3DOF Haptics
  - Spatial Partitioning
    - Uniform grid implemented using hashing
  - Bounding Volume Hierarchy
    - Oriented bounding box trees
  - Frame-to-Frame Coherence
    - Caching the last "witness"

Gregory, Gottschalk, Taylor & Lin
[VR’99, CGTA’00]

HCOLLIDE Overview (I)

- OFFLINE PROCESS
  - Pre-compute hybrid representation, consisting of uniform grids and each contains an OBBTree.

- RUNTIME PROCESS
  - Identify "contact region" by uniform spatial partitioning (implemented with hash table)
  - Locate the exact contact points by querying and traversing the OBBTrees
  - Frame-to-frame coherence by caching the last "witness"
  - Find the projected surface contact point

H-Collide Overview (II)

Spatial partitioning with hash table
Ray Vs. OBBTree test
SCP computation
SCP

HCOLLIDE Pseudo Code

\[
\begin{align*}
X &= |w_x| \\
Y &= |w_y| \\
Z &= |w_z| \\
\text{if } |m_x| > X + t_x \text{ return disjoint} \\
\text{if } |m_y| > Y + t_y \text{ return disjoint} \\
\text{if } |m_z| > Z + t_z \text{ return disjoint} \\
\text{if } |m_x| w_z - m_z w_x| > t_x Z + t_z Y \text{ return disjoint} \\
\text{if } |m_y| w_z - m_z w_y| > t_y Z + t_y X \text{ return disjoint} \\
\text{if } |m_z| w_x - m_x w_z| > t_z Y + t_z X \text{ return disjoint}
\end{align*}
\]
Specialized Overlap Test

- Simple control loop
  - good for micro-coding & SIMD implementation
- Cost: 42-72 arithmetic operations
  - 9 absolute values
  - 6 comparisons
  - 9 addition/subtraction
  - 12 multiplication
  - 36 ops for transformation

HCOLLIDE Hashing

- Hashing Function:
  \[ h(k) = x + y \cdot \text{num\_cell} + z \cdot \text{(num\_cell)}^2 \]
  \[ \text{TableLoc} = \text{random}(h(k)) \mod \text{TableLength} \]
- Grid Size Selection:
  - difficult to compute an optimal value for all input models with varying triangulation
  - set the grid size to be the \textit{averaged} edge length of the input model

Optimal Grid Size (I)

\textbf{Assume --}

- Line segment swept out by the probe is small compared to the optimal grid size
- There is only one contacting point with the surface of the object and one triangle in contact with the probe
- The triangulation of the object is uniform & all triangles have nice aspect ratio
- All objects in the scene are static and rigid

Optimal Grid Size (II)

- \( N \): total number of triangle
- \( M \): averaged number of triangles per cell
  \[ C_r = (2 \log N + 1) \cdot C_{obb} + C_{tri} \]
  \[ C_g = M \cdot C_{tri} + C_l \]
  \[ C_h = (2 \log M + 1) \cdot C_{obb} + C_{tri} + C_l \]
  \[(2\log M \cdot C_{obb}/C_{tri} + 1 + C_{obb}/C_{tri}) < M < N/2^{C_{obb}/C_{tri}} \]

According to our implementation:
  \( C_l/C_{obb}: 0.9-5.5 \text{ and } C_{obb}/C_{tri}: 0.764-4.0 \)

Nano-Surfaces

HCOLLIDE: Timing on Nano-Surfaces (msec)

<table>
<thead>
<tr>
<th>Method</th>
<th>Hash Grid</th>
<th>Hybrid</th>
<th>OBB Tree</th>
<th>Ghost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ave Col. Hit</td>
<td>0.0138</td>
<td>0.0101</td>
<td>0.0134</td>
<td>0.332</td>
</tr>
<tr>
<td>Worst Col. Hit</td>
<td>0.125</td>
<td>0.168</td>
<td>0.0663</td>
<td>0.724</td>
</tr>
<tr>
<td>Ave Col. Miss</td>
<td>0.00739</td>
<td>0.00508</td>
<td>0.00422</td>
<td>0.0109</td>
</tr>
<tr>
<td>Worst Col. Miss</td>
<td>0.0347</td>
<td>0.0377</td>
<td>0.0613</td>
<td>0.210</td>
</tr>
<tr>
<td>Ave Int. Hit</td>
<td>0.0428</td>
<td>0.0386</td>
<td>0.0447</td>
<td>0.0851</td>
</tr>
<tr>
<td>Worst Int. Hit</td>
<td>0.0877</td>
<td>0.102</td>
<td>0.0690</td>
<td>0.175</td>
</tr>
<tr>
<td>Ave Int. Miss</td>
<td>0.0268</td>
<td>0.0197</td>
<td>0.0213</td>
<td>0.0545</td>
</tr>
<tr>
<td>Worst Int. Miss</td>
<td>0.0757</td>
<td>0.0697</td>
<td>0.0587</td>
<td>0.284</td>
</tr>
<tr>
<td>Ave. Query</td>
<td>0.022</td>
<td>0.016</td>
<td>0.039</td>
<td>0.18</td>
</tr>
</tbody>
</table>
Ford Bronco

Butterfly

HCOLLIDE: Algorithm Analysis

• At least 2-20 times faster than GHOST on the models we have tested on

• Hybrid is the most favorable and capable of maintaining kHZ rate

• If the number of triangles per grid cell is relatively small compared to the model size, then hybrid method runs in constant time

HCOLLIDE: Timing on Ford Bronco (msec)

<table>
<thead>
<tr>
<th>Method</th>
<th>Hash Grid</th>
<th>Hybrid</th>
<th>OBB Tree</th>
<th>Ghost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ave Col. Hit</td>
<td>0.0113</td>
<td>0.00995</td>
<td>0.0125</td>
<td>0.104</td>
</tr>
<tr>
<td>Worst Col. Hit</td>
<td>0.136</td>
<td>0.132</td>
<td>0.177</td>
<td>0.495</td>
</tr>
<tr>
<td>Ave Col. Miss</td>
<td>0.0133</td>
<td>0.00731</td>
<td>0.0189</td>
<td>0.0280</td>
</tr>
<tr>
<td>Worst Col. Miss</td>
<td>0.128</td>
<td>0.0730</td>
<td>0.137</td>
<td>0.641</td>
</tr>
<tr>
<td>Ave Int. Hit</td>
<td>0.0566</td>
<td>0.0374</td>
<td>0.609</td>
<td>0.0671</td>
</tr>
<tr>
<td>Worst Int. Hit</td>
<td>0.145</td>
<td>0.105</td>
<td>0.170</td>
<td>0.293</td>
</tr>
<tr>
<td>Ave Int. Miss</td>
<td>0.0523</td>
<td>0.0223</td>
<td>0.0452</td>
<td>0.0423</td>
</tr>
<tr>
<td>Worst Int. Miss</td>
<td>0.132</td>
<td>0.133</td>
<td>0.167</td>
<td>0.556</td>
</tr>
<tr>
<td>Ave. Query</td>
<td>0.027</td>
<td>0.014</td>
<td>0.028</td>
<td>0.048</td>
</tr>
</tbody>
</table>

HCOLLIDE: Timing on Butterfly (msec)

<table>
<thead>
<tr>
<th>Method</th>
<th>Hash Grid</th>
<th>Hybrid</th>
<th>OBB Tree</th>
<th>Ghost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ave Col. Hit</td>
<td>0.0232</td>
<td>0.0204</td>
<td>0.0163</td>
<td>1.33</td>
</tr>
<tr>
<td>Worst Col. Hit</td>
<td>0.545</td>
<td>0.198</td>
<td>0.100</td>
<td>5.37</td>
</tr>
<tr>
<td>Ave Col. Miss</td>
<td>0.00896</td>
<td>0.0405</td>
<td>0.00683</td>
<td>0.160</td>
</tr>
<tr>
<td>Worst Col. Miss</td>
<td>0.237</td>
<td>0.139</td>
<td>0.121</td>
<td>3.15</td>
</tr>
<tr>
<td>Ave Int. Hit</td>
<td>0.228</td>
<td>0.0659</td>
<td>0.0704</td>
<td>0.509</td>
</tr>
<tr>
<td>Worst Int. Hit</td>
<td>0.104</td>
<td>0.138</td>
<td>0.103</td>
<td>1.952</td>
</tr>
<tr>
<td>Ave Int. Miss</td>
<td>0.258</td>
<td>0.0279</td>
<td>0.0256</td>
<td>0.229</td>
</tr>
<tr>
<td>Worst Int. Miss</td>
<td>0.0544</td>
<td>0.131</td>
<td>0.0977</td>
<td>3.28</td>
</tr>
<tr>
<td>Ave. Query</td>
<td>0.030</td>
<td>0.016</td>
<td>0.016</td>
<td>0.32</td>
</tr>
</tbody>
</table>

Applications: inTouch

• Direct Haptic Interaction
• Multiresolution Modeling
• 3D Painting on Polygonal Meshes

Gregroy, Ehmann, Lin [VR 2000]
**inTouch: System Architecture**

- **Haptic Device**
- **Network**
- **Haptic Server**
- **H-Collide**
- **GHOST**
- **Multiresolution Mesh**
- **Client Application**
- **Graphical Display**
- **Painting/Modeling UI**
- **Multiresolution Mesh**

**inTouch: User Interface**

**inTouch: Examples**

- Painted Butterfly (~80k triangles)

**ArtNova: Touch-Enabled 3D Model Design**

- Interactive texture painting
- User-centric viewing
- Realistic force response

**http://gamma.cs.unc.edu/ArtNova**

**Foskey, Otaduy & Lin [VR 2000]**

**6-DOF Haptic Display Using Localized Computations**

- Decompose objects into convex pieces and compute a set of localized pairwise PD's
- Use dual-space expansion to quickly estimate the PD between convex polytopes
- Cluster nearby surface contacts for localized force computation based on PD estimates and predictive methods

**http://gamma.cs.unc.edu/6DOFLCC/**

**Kim, Otaduy, Lin & Manocha [HS 2002]**

**http://www.cs.unc.edu/~geom/inTouch**

**http://www.cs.unc.edu/~geom/inTouch**
Collision Detection – SWIFT++

- A fast collision detection library using bounding volume hierarchies of convex hulls
- The overlap test between two convex bounding boxes is performed using fast Voronoi Marching
- When collisions occur, needs penetration depth

http://gamma.cs.unc.edu/SWIFT++
Ehmann & Lin [Eurographics 2001]

Penetration Depth (PD)

- Minimum translational distance to make \( P \) and \( Q \) disjoint over all possible directions.
- Minimum distance from origin \( O_{\text{diff}} \) to surface \( \partial(P-Q) \) of Minkowski difference.

Gauss Map

- Mapping from a feature \((V,E,F)\) in 3D to a unit sphere, according to surface normal
  - Face (F) Point in Gauss map
  - Edge (E) Great arc in Gauss map
  - Vertex (V) Region bounded by great arcs

- The Minkowski sum of two convex objects is computed from the overlay of their Gauss maps.

Overview of the Algorithm

- Precomputation
  - Decomposition of an object into convex pieces using the surface decomposition.
- Runtime
  1. Intersection Test
     - Hierarchical pairwise test for intersection.
     - Identify intersecting convex pieces.
  2. PD Computation
     - Walk on the surface of the Minkowski difference of the intersecting convex pieces by minimizing the distance from the origin to the surface.

Minkowski Sum/Difference

- Minkowski Sum:
  \[ P+Q = \{p+q | p \in P, q \in Q \} \]
- Minkowski Difference:
  \[ P-Q = \{p-q | p \in P, q \in Q \} \]
  Negate \( Q \) and compute \( P+(-Q) \)

Minkowski Difference from Gauss Map

- Gauss maps of the intersected convex pieces
- Overlaid and projected Gauss maps
- Overlay region that implicitly defines the Minkowski difference

FV, VF and EE pairs correspond to the vertices of the Minkowski difference.
**Incremental Search of PD**

- At a certain vertex in the overlay, check its corresponding PD with that of its neighbors. March toward that vertex that minimizes the PD.
- The actual Minkowski difference is locally computed when needed.

**Initialization**

- Centroid difference vector
- Penetration witness features
- Motion coherence

- Use the one that minimizes PD

**Possible Problems**

- Local minima: distance function from the origin to the surface of Minkowski difference can have multiple local minima. Good initialization provides a good result.
- Degeneracies:
  - Coplanar faces. Mapped to the same point in Gauss map. Treat them as a single point, and join the neighbors.
  - Central projection of Gauss map. Solved by local computation at each iteration.

**Extension to Non-convex Objects**

- Pairwise computation of PD.
- Problems originated from surface convex decomposition:
  - Convex pieces completely penetrating the other object
  - PD returns a "virtual feature" that does not exist in the original model
  - We circumvent the problem by traversing to the neighboring features.

**Contact Clusters**

- Variable number of contacts translates into a variable stiffness.
- Cluster contacts based on the distance between them.
- Compute a new contact (PD, normal, application point) as a weighted average, where weight = PD.
- In practice, the force output is smooth.

**Results**

- Video
Results

For more information, see
http://gamma.cs.unc.edu/interactive
http://gamma.cs.unc.edu/HCollide
http://gamma.cs.unc.edu/inTouch
http://gamma.cs.unc.edu/ArtNova
http://gamma.cs.unc.edu/6DOFLCC/

The End