Collision Detection I
Survey January/February 2004

Admittance-based devices, such as the Haptic Master, reduce impedance-type architectures are most common. Position. Simpler to design and much cheaper to simulate mechanical admittance—they read force and send objects populating the environment. Moreover, haptic-presentation inside the virtual environment and the virtual interaction forces between the haptic interface represent the ideal interaction force to the best of the device's capabilities. We consider haptic rendering algorithms applicable splitting haptic rendering into three main blocks. Haptic interface devices include between device and operator. A DOF can be passive or body interface—that is, the number of dimensions characterized in a large workspace. Characteristics commonly considered desirable for haptic devices simulating mechanical impedance—they read contact area, and so on) have occurred. Force-response algorithms to the user. Control algorithms command the exact force computed by the force-response algorithms to the user. Control algorithms compute the correct contact forces that would normally arise during reducing overall performance. Hardware limitations prevent haptic devices from returning contact forces that would normally arise during reducing overall performance. Their return values are normally force and torque vectors so users don't have to unconsciously compensate for parasitic forces; frequency properties (thereby regularizing the device operation); low back-drive inertia and friction; minimal constraints on motion imposed by the device. Balanced range, resolution, and bandwidth of position and force between device and operator. A DOF can be passive or body interface—that is, the number of dimensions characterized. Another distinction between haptic interface devices is their intrinsic mechanical behavior. Among different avatars. For example, a surgical tool is directly rendered such forces on the human operator. Haptic-rendering algorithms ensure that the haptic device corresponds to the haptic-rendering algorithms often makes it impossible the contact forces that would normally arise during reducing overall performance. Their return values are normally force and torque vectors so users don't have to unconsciously compensate for parasitic forces; frequency properties (thereby regularizing the device operation); low back-drive inertia and friction; minimal constraints on motion imposed by the device.
Problem Definition

- We seek efficient algorithms to answer the following queries:
  - Intersection query (do objects overlap?)
  - Contact manifolds (set of contact points)
  - Penetration depth / intersection volume
  - Separation distance
- Difficulty increases as we move down...
Geometric Representations

Many different ways to describe the same object
Surface Representations

- Implicit surface: $S(x, y, z) = 0$
- Parametric surface: $P(u, v) | u, v \in \mathcal{D}$
- Point-sampled surface (point cloud)
- Polygonal mesh:
  - Triangle mesh
  - Quadrilateral (quad) mesh
- ... any other ones you can think of?
Triangle Meshes

Why is this the most popular representation?
Terminology

- Objects are composed of primitive shapes

- **Broad phase**
  - Which objects are in a vicinity?

- **Narrow phase**
  - Does the geometry intersect?
Broad Phase Collisions

No possibility of intersection
Narrow Phase Collisions

Collision!

CS277 - Experimental Haptics, Stanford University, Spring 2014
Today’s Lecture
Primitive Tests
Tests for Meshes

- The two most common collision queries for haptic rendering of polygonal meshes:
  - line segment-triangle intersection test
  - triangle-triangle intersection test
Ray-Triangle Intersection

- Find intersection between line and plane
- Discard if point is outside segment range
- Use barycentric coordinates to determine if the point is inside the triangle
Barycentric Coordinates

$$f(u, v) = (1 - u - v)p_0 + up_1 + vp_2$$
Barycentric Coordinates

\[ f(u, v) = (1 - u - v)p_0 + up_1 + vp_2 \]

\[ u = \frac{A_1}{A} \]

\[ v = \frac{A_2}{A} \]

\[ A = \frac{1}{2} \left| (p_1 - p_0) \times (p_2 - p_0) \right| \]
A Direct Approach

A ray: \( r(t) = o + td \)

A triangle: \( f(u, v) = (1 - u - v)p_0 + up_1 + vp_2 \)

Ray-triangle intersect: \( o + td = (1 - u - v)p_0 + up_1 + vp_2 \)

Rearrange terms: \[
\begin{pmatrix}
-d & p_1 - p_0 & p_2 - p_0
\end{pmatrix}
\begin{pmatrix}
t \\
u \\
v
\end{pmatrix} = o - p_0
\]

Solve for \( t, u, \) and \( v \) ...
Cramer’s Rule

- Given the set of linear equations

\[
\begin{pmatrix}
a & b & c \\
a & b & c \\
a & b & c
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z
\end{pmatrix}
= d
\]

- Write the determinant of the matrix

\[
\text{det}(a, b, c) = \begin{vmatrix}
a_1 & b_1 & c_1 \\
a_2 & b_2 & c_2 \\
a_3 & b_3 & c_3
\end{vmatrix}
\]

- Then

\[
x = \frac{\text{det}(d, b, c)}{\text{det}(a, b, c)} \quad y = \frac{\text{det}(a, d, c)}{\text{det}(a, b, c)} \quad z = \frac{\text{det}(a, b, d)}{\text{det}(a, b, c)}
\]
A Direct Approach

Our equation:

\[
\begin{pmatrix}
-d & p_1 - p_0 & p_2 - p_0
\end{pmatrix}
\begin{pmatrix}
t \\
u \\
v
\end{pmatrix} = o - p_0
\]

Applying Cramer's rule:

\[
\begin{pmatrix}
t \\
u \\
v
\end{pmatrix} = \frac{1}{\det(-d, e_1, e_2)}
\begin{pmatrix}
\det(s, e_1, e_2) \\
\det(-d, s, e_2) \\
\det(-d, e_1, s)
\end{pmatrix}
\]

where \( e_1 = p_1 - p_0, \ e_2 = p_2 - p_0, \ s = o - p_0 \)

Check \( t, u, v \) within intervals!
Geometric Interpretation

\[ M = \begin{pmatrix} -d & p_1 - p_0 & p_2 - p_0 \end{pmatrix} \]

Triangle-Triangle Intersection

- Triangles A and B may intersect if they cross each other’s plane
- Test A’s vertices against B’s plane, and vice versa for rejection
- Test for interval overlap along the line of intersection
Half-Plane Test

\[ [a, b, c, d] = \begin{vmatrix} a_x & b_x & c_x & d_x \\ a_y & b_y & c_y & d_y \\ a_z & b_z & c_z & d_z \\ 1 & 1 & 1 & 1 \end{vmatrix} \]

\[ = (d - a) \cdot ((b - a) \times (c - a)) \]

- Geometric interpretation:
  - This tests which side of the plane defined by triangle abc the point d is on
Half-Plane Test

- Given two triangles:
  \[ \triangle p_1 q_1 r_1 \text{ and } \triangle p_2 q_2 r_2 \]

- We can first perform the half-plane test on triangle one:
  \[
  [p_2, q_2, r_2, p_1] \quad [p_2, q_2, r_2, q_1] \quad [p_2, q_2, r_2, r_1]
  \]

- Then symmetrically perform the half-plane test on the other triangle...
Intersection of Intervals

Interval Intersection Test

- Intervals on line L are
  \[ I_1 = [i, j] \quad I_2 = [k, l] \]

- Intervals overlap if
  \[ k \leq j \quad \text{and} \quad i \leq l \]

- Perform two additional determinant tests:
  \[ [p_1, q_1, p_2, q_2] \]
  \[ [p_1, r_1, r_2, p_2] \]
Triangle-Triangle Summary

- Compute three 4x4 determinants to test first triangle against the second’s plane
- If triangle intersects the plane, perform the symmetric test using three more determinants
- If both triangles intersect the other’s plane, perform interval overlap test on the intersecting line with two last 4x4 determinants
- What happens with co-planar triangles?!
Two of My Favorite Books

No need to memorize any of these algorithms!
Some Easier Stuff...

- Mesh geometry intersection tests are expensive, and must be performed for every triangle
- Collision detection can be sped up significantly by using rejection tests on bounding volumes
Bounding Volumes

- Most common bounding volumes are **spheres** and **boxes**
- Two most common collision queries:
  - Sphere-sphere intersection
  - Box-box intersection
Sphere-Sphere Intersection

Easiest one in the book!
Sphere-Sphere Intersection

- Two spheres intersect if the separation between their centers is less than the sum of their radii:

\[ \| \mathbf{c}_1 - \mathbf{c}_2 \| < r_1 + r_2 \]
Box-Box Intersection

- An axis-aligned box is represented by lower (minimum coordinate) and upper (maximum coordinate) vertices
- How do we detect intersection of boxes?
Box-Box Intersection

- Two axis-aligned boxes intersect if the lower coordinate of each box is bounded by the upper coordinate of the other:

\[ a_{\text{min}} < b_{\text{max}} \]
\[ b_{\text{min}} < a_{\text{max}} \]
Oriented Box Intersection

How do we test for this kind of box intersection?
Separating Hyperplane Theorem

- Two convex polytopes can be separated by a hyperplane if and only if they are disjoint.
- For disjoint polyhedra, there exists a separating plane parallel to a face on either polyhedron, or an edge selected from each polyhedron (why?)
Separating Axis Test

- Project all vertices onto the normal of the separating plane ("separating axis")
- Projections from each polytope form an interval
- Polytopes are disjoint if intervals are disjoint
Oriented Box Intersection
Oriented Box Intersection

- Perform separating axis test on every possible axis:
  - 3 axes (faces of box A)
  - 3 axes (faces of box B)
  - $3 \times 3 = 9$ axes from pairs of box edges

- Total: 15 separating axis tests
Convex Polyhedra

How would we test shapes like these?
Intersection of Convex Polyhedra

- Use the separating hyperplane theorem:
  - How do we determine what the plane/axis is?

- Use a feature tracking approach:
  - Polyhedra will collide when their separation distance vanishes
  - Can we tell which elements are about to collide?
Linear Programming

- Maximize

\[ c^T x \]

- Subject to the linear constraints

\[ Ax \leq b \]

- where \( x \) is a vector of \( n \) unknowns, \( c \) is a vector of coefficients, and \( A \) and \( b \) are constraints which define an \( n \)-dimensional convex polytope

- Can detect infeasibility
Linear Programming

- Four coefficients of the separating plane are linear programming variables

- Constrain all vertices of polyhedron A to one side of plane and vertices of polyhedron B to the other

- LP tells us whether or not a separating plane exists

- Expected linear time

\[ ax + by = c \]
\[ (a, b, c) = ? \]

\[ ax + by - c > 0 \]

\[ ax + by - c < 0 \]
Closest Feature Tracking

- Proposed by Lin & Canny (1991)

- Observe that when contact occurs, it will occur between the two closest points

- Two contact configurations provide features that determine the closest points

- Can use Voronoi regions to track the closest features on non-overlapping convex polyhedra
Voronoi Regions

- Defines a convex region of space for each feature containing the set of points closer to it than any other feature.
Feature Tracking

Walk along outside as objects change position...
Lin-Canny Algorithm

- Computes minimum separation distance between closest pair of features
- Takes advantage of *temporal coherence*
- What is the expected running time of this algorithm?
Lin-Canny Algorithm

- Performance can be near **constant** time once the tracking is initialized
- Can be quite difficult to implement!

![Figure 1: Point-Vertex Applicability Criterion](image1.png)
![Figure 2: Point-Edge Applicability Criterion](image2.png)
![Figure 3: Vertex-Face Applicability Criterion](image3.png)


CS277 - Experimental Haptics, Stanford University, Spring 2014
Primitive Test Summary

- We covered fast intersection tests for:
  - Segment-triangle
  - Triangle-triangle
  - Sphere-sphere
  - Axis-aligned bounding box (AABB)
  - Oriented bounding box (OBB)
  - Convex polyhedra

- Look up others if needed!
Summary

‣ We can test collision between two objects, but what happens if there are thousands?