

Haptic rendering CY4J9 W.S. Harwin

See <http://www.rdg.ac.uk/~shshawin/LN> for pdf version

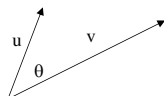
- Identical concepts to visual rendering
- Usually includes other modalities, in particular vision
- Stability criteria require fast update speeds typical 500-1000Hz

Phases to haptic rendering

- Collision detection
 - Open source eg I-Collide, Solid, opcode, gimpact
 - Haptic specific (must be fast and efficient)
 - Only a polygon/point collision algorithm is required.
- Force estimation
- Collision response
 - Requires a model of what should happen, eg physics engine

Dot product

- Given two vectors u and v can work out a scalar dot product
- $$u \cdot v = u_1 v_1 + u_2 v_2 + u_3 v_3 = |u||v|\cos \theta$$
- If one vector is a unit vector, then dot product represents the projection of the second vector onto the first

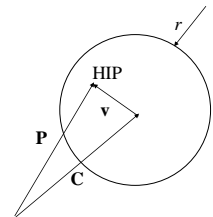


Definitions

- PHIP = Physical haptic interface point
- HIP = haptic interface point
- SP = surface contact point
- GO = god-object (where the HIP should be)

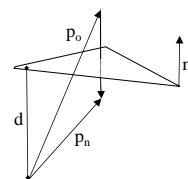
Sphere Collision

- v is vector from Centre of sphere C to Haptic point P
- $v = P - C$
- If $(v^T v < r^2)$ then HIP is in the sphere
- Force response eg $F = K (r - |v|)v / |v|$



Collision by polygon transition

- d is (scalar) perpendicular distance to polygon from origin
- n is outward facing normal
- Distance from origin along perpendicular is $p_o \cdot n$ and $p_n \cdot n$



Condition for collision of line with polygon

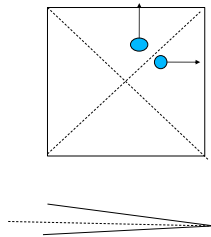
- $(n \cdot p_n - d) > 0$ and $(n \cdot p_o - d) < 0$ or visa verse
- Calculate as $(n \cdot p_n - d) (n \cdot p_o - d) < 0$
- If this is true the 'ray' has passed through the plane of the polygon
- Must then check with three planes forming a 'toblerone bar' around this triangle to see if ray penetrates into the bar and hence the polygon

Force estimation

- Hookes law $F = Kq$
- Visco elastic $F = Kq + B \, dq/dt$
- Fluid $F = B \, dq/dt$

Simplistic collision response

- evaluate q as the perpendicular distance to nearest surface. Eg cube
- $F = Kq$
- Problem with push through of narrow objects



Zilles and Salisbury god-object

- Maintains a history of contact
- God-object is connected to HIP by a virtual spring
- God object is
 - either coincident with HIP (freespace)
 - or on the surface of the object while HIP is in object

God-object algorithm

- Algorithm needs to track god-object on
 - A) surface of entry polygon
 - B) transition to edge or vertex of joining polygons
 - C) surface of additional polygons
 - D) transition into free space

Zilles and Salisbury algorithm

If polygon is defined by $Ax + By + Cz = D$

Surface normal is $\left[\frac{A_1}{D_1}, \frac{B_1}{D_1}, \frac{C_1}{D_1} \right]$

For the closest point to a surface form

$$A = \begin{bmatrix} A_1 & B_1 & C_1 \end{bmatrix}, D = \begin{bmatrix} D_1 \end{bmatrix}$$

For the closest point to an edge (two intersecting polygons) form

$$A = \begin{bmatrix} A_1 & B_1 & C_1 \\ A_2 & B_2 & C_2 \end{bmatrix}, D = \begin{bmatrix} D_1 \\ D_2 \end{bmatrix}$$

Zilles and Salisbury

For the closest point to a vertex (three intersecting polygons) form

$$A = \begin{bmatrix} A_1 & B_1 & C_1 \\ A_2 & B_2 & C_2 \\ A_3 & B_3 & C_3 \end{bmatrix}, D = \begin{bmatrix} D_1 \\ D_2 \\ D_3 \end{bmatrix}$$

Solution is

$$\underline{x}_{go} = \underline{x}_p + A(A^T A)^{-1} (D - A^T \underline{x}_p)$$

Where x_p is current phantom position and x_{go} is the position of the god-object on the polygon

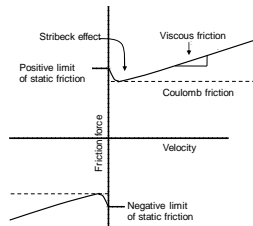
Picking up an object

- In Zilles and Salisbury algorithm friction is then added in if needed.
- Friction is needed to form a stable grasp
- Good friction model needed to lift against gravity

Friction Background - Friction

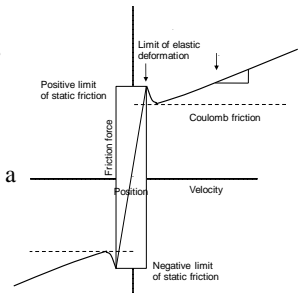
Classical Friction Model:

- Typically has response as shown.
- LeGru model includes model of micro-slip behaviour
- Simpler models are Karnopp and Dahl.
- Coulomb model assumes no viscous friction.



Bristle model of friction

- Analogous to object sliding on the bristles of a brush
- Stiffness defined across elastic range, thereafter response is a function of velocity



The Dahl and LuGre Model

Dahl Friction model

$$\frac{dF}{dx} = \sigma_0 \left(1 - \frac{F}{F_v} \text{sign}(v) \right)$$

On integration this becomes

$$F = F_c (1 - e^{-\sigma_0 |x| / F_c}) \text{sign}(v)$$

LuGre Friction Model. The State equations are

$$\dot{z} = v - h(v)z$$

$$F = \sigma_0 z + \sigma_1 \dot{z} + f(v)$$

where $h(v) = \sigma_0 \frac{|v|}{g(v)}$

Typically $f(v)$ would be viscous friction, i.e. $f(v) = \sigma_2 v$. A reasonable choice for $g(v)$ that models both coulomb friction and the Stribeck effect is

$$g(v) = F_c + (F_s - F_c) e^{-|v/v_s|^\alpha}$$

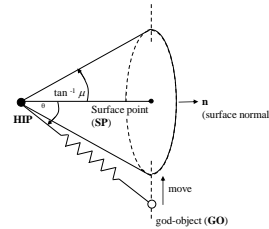
The Dahl and LuGre Model

- ▶ The Dahl model is a simplified version of the LuGre model
- ▶ The Dahl model does not account for the stribeck effect
- ▶ The state z , can be considered as a bristle deflection
- ▶ F_c is the coulomb friction F_s is the limit of static friction
- ▶ In the LuGre model typical values are $0.5 < \alpha < 2$

Friction cone algorithm (Melder and Harwin)

- Friction Cone Algorithm is active only when the haptic interaction point is inside an object.
- A cone is placed at the haptic interaction point oriented in the direction of the normal of the contacted surface.
- Intersection of this cone on the surface defines a friction circle.
- Define $\mu = \tan \theta = \frac{F_{friction}}{R}$

Friction Cone Algorithm



- GO – HIP α Force Vector to apply to Haptic Device
- Since Vector, will work in any number of dimensions (ie. 2D, 3D, 4D)

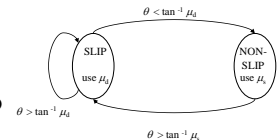
Calculating friction

- Calculate depth of penetration $d = (\mathbf{HIP} - \mathbf{GO}) \cdot \mathbf{n}$.
- Hence, calculate. $\mathbf{SP} = \mathbf{HIP} + d\mathbf{n}$.
- Circle (the intersection of the friction cone with the surface polygon) has radius $R = d\mu$ (μ is the friction coefficient for the surface).
- Compare distance between the surface point (SP) and the current god-object (GO) i.e. $r = |\mathbf{GO} - \mathbf{SP}|$ to the radius of the friction circle. Update the god-object if outside circle.

$$\mathbf{GO}_{new} = \mathbf{SP} + R \cdot \frac{(\mathbf{GO} - \mathbf{SP})}{r}$$
- The response force can now be calculated based upon the vector from the HIP to the god-object and will be proportional to the surface stiffness.

Modeling dynamic (coulomb) and static friction

- State transition diagram – either slipping or not
- If slipping use coulomb μ
- If GO within friction circle change state and use static μ

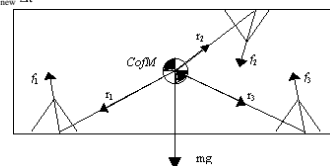


Manipulation of Virtual Objects

- With multi-point haptics, require a method to allow for realworld/natural manipulation of virtual objects.
- Direction vector from HIP to GO is proportional to force applied by haptic interface to object.
- Summing all force vectors on the object gives the residual force in the object (including gravity etc.)
- Residual force is input to a suitable movement algorithm.
- 2 Types of manipulation – Translation & Rotation.

Residual Force Algorithm

- Sum of Forces in virtual object required for translation..
- Object requires a mass.
- Point Mass model used ie. all forces act on the Centre of Mass.
- Residual Force in Object = Mass x Acceleration
- $\mathbf{v}_{new} = \mathbf{v}_{old} + \mathbf{a} \Delta t$ (where Δt = loop time of the control algorithm)
- $\mathbf{p}_{new} = \mathbf{p}_{old} + \mathbf{v}_{new} \Delta t$

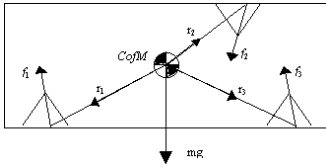


Residual Torque Algorithm

- Sum of Torques in virtual object required for rotation.
- Object requires inertias along each of the primary axes.

• Inertia stored in Matrix form ie, in 3D $I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$

- Torque = $r \times f$



Residual Torque Algorithm

- Residual Torque in object: $T = \sum_{i=1}^n r_i \times f_i$
- Angular acceleration given by: $\dot{\omega} = {}^J J^{-1} (T - \omega \times {}^J J \omega)$ where ${}^J J = R J_p R^T$
- Can simplify and use: $\dot{\omega} \approx R J_p^{-1} R^T T$
- Angular Velocity and Rotation Estimate is given by:

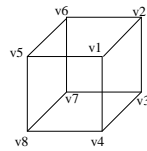
$$\begin{aligned} \omega &= \omega_{old} + \dot{\omega} \Delta t \\ \Delta R &= I + S \Delta t \\ R &= R_{old} \Delta R \end{aligned} \quad \text{where } S = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

Haptic Rendering of Complex Shapes

Convexity & Concavity

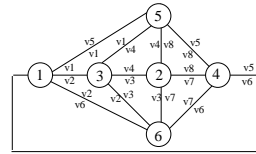
- A Polygon is convex if all internal angles $< 180^\circ$.
- A Polygon is concave if any internal angle $> 180^\circ$.
- An edge is convex if the internal angle between the polygons $< 180^\circ$.
- A vertex
 - Totally convex if all the adjoining edges are convex.
 - Totally concave if all the adjoining edges are concave.
 - Partially convex/concave if the edges are mixed (ie. concave and convex).

Face Directed Connection Graph



- F1 = Top face
- F2 = Bottom face
- F3 = Right face
- F4 = Left face
- F5 = Front face
- F6 = Back face

- Made from Nodes, Connections & corners.
- Node contains:
 - Face plane
 - Array of Connections
- Connection:
 - SharedVertices
 - ConnectedNodes
 - EdgeVector
 - ConnectionType
 - VoronoiPlanes
 - EndPlanes
- Corner Contains:
 - Vertex
 - Array of Connections

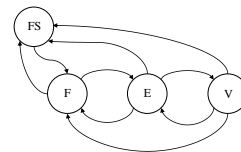


(Pseudo) Voronoi Regions and End Planes

- A feature's Voronoi region is an area of space around the feature that is closest to that feature.
- Each triangular face has 3 associated voronoi planes that pass through each of the face's edges perpendicular to the face.
- Each edge has 2 associated end planes, that pass through each of the edge's vertices perpendicular to the edge.

State Transitions

- FS - F determined by collision algorithm.



FS = Free-space F = Face E = Edge V = Vertex

Needle sticking, cutting, wearing, stretching

- Part of the collision response/physics
- Represent solid objects
 - Nested surface (triangulated)
 - Tetrahedra or cubic meshes
 - Spherical meshes
- Needle stick needs to simulate membrane transition, allow GO to drop through a nested set of surfaces
- Cutting – As vector between HIP and GO transitions separate the mesh connectivity

Needle sticking, cutting, wearing, stretching

- Wearing
 - Remove tetrahedra/cube/sphere after a time in contact
- Stretching
 - Finite element methods (not realistic with current computation)
 - Linear Spring/mass/damper networks linking the individual nodes.
 - Simplified tetrahedra physics.

References

- <http://www.cyber.rdg.ac.uk/ISRG/haptics/>
- N. Melder, W. S. Harwin and P. M. Sharkey, Translation and Rotation of Multi-Point Contacted Virtual Objects, *Proceedings of Eurohaptics Conference*, 2003
- W. S. Harwin and N. Melder, Improved Haptic Rendering for Multi-Finger Manipulation Using Friction Cone based God-Objects, *Proceedings of Eurohaptics Conference*, 2002
- C. B. Zilles and J. K. Salisbury, A Constraint-based God-object Method for Haptic Display, *Proceedings of International Conference on Intelligent Robots and Systems*, 1995
- C. Ho, C. Basdogan, M. A. Srinivasan, Efficient Point-Based Rendering Techniques for Haptic Display of Virtual Objects, *Presence* 8(5) Oct 1999 pp 477-491