08: CONTACT MODELLING

15/02/2016
The purpose of Narrow Phase: Exact collision detection **AND** also where precisely have objects touched, where to apply forces, torques?
LOOKAHEAD: COLLISION RESPONSE

The following is an equation for Rigid Body Impulsive collision response (see Baraff et al, we will discuss in detail later)

\[ j = \frac{1}{M_a} + \frac{1}{M_b} + \hat{n}(t_0) \cdot (I_a^{-1}(t_0) (r_a \times \hat{n}(t_0))) \times r_a + \hat{n}(t_0) \cdot (I_b^{-1}(t_0) (r_b \times \hat{n}(t_0))) \times r_b \]

\[ I^{-1}(t) = R(t) I_{body}^{-1} R(t)^T \]

\[ \dot{r}_a = p - x_a(t_0) \quad \text{and} \quad r_b = p - x_b(t_0) \]

\[ v_{rel} = \hat{n}(t_0) \cdot (\dot{p}_a(t_0) - \dot{p}_b(t_0)) \]

\[ \dot{p}_a(t_0) = v_a(t_0) + \omega_a(t_0) \times (p_a(t_0) - x_a(t_0)) \]

\[ \dot{p}_b(t_0) = v_b(t_0) + \omega_b(t_0) \times (p_b(t_0) - x_b(t_0)) \]

\[ p \text{ is the contact point associated with a contact direction } n \]
MODELLING CONTACT

- At point of contact two smooth surfaces will have coinciding tangent planes: a plane of contact
- The normal of this plane is the **contact normal**
- The point on each object that we need to apply the appropriate force is the **point of contact**
INEXACT CONTACT

**In ideal situations:** a single contact point usually where distance between objects is zero i.e.

\[ p = p_A = p_B \]

**In practice:** the distance is slightly non-zero due to numerical (floating point) imprecision and timestep limitations.

- Normally consider a collision envelope
  \[ \varepsilon = +/– 10^{-7} \text{ or } 10^{-15} \]
- We can take the mean of the closest points as the point of collision
  \[ p = \frac{p_A + p_B}{2} \]
CONTACT MANIFOLDS

We rarely have one idealised point of contact

Simulation often leads to more than just a point of contact - more likely a surface or contact manifold
Types of Contact

Three features V, E, F (vertex, edge, face) lead to six types of contacts:

- V-V
- V-E
- V-F
- E-E
- E-F
- F-F
Let us assume

- that we are usually dealing with polygonal models
- that we know the two closest features on two objects

Contacts are identified with respect to features - one from each polyhedra

A principal Contact a.k.a. Point of contact $p$ is computed based on the closest points between the two colliding features

- On each of the features we want to evaluate a closest point to the colliding feature on the opposite object
CLOSEST POINTS DETERMINATION

**V-V Collision**
- Take vertices as closest features
- Q: What is the collision plane/normal?

**For two vertices** $p_1, p_2$:
- Distance is: $d = |p_1 - p_2|$
- Closest point is midway between the two vertices:
  $$p = \frac{p_1 + p_2}{|p_1 - p_2|}$$
- Normal can be taken as the line between the two:
  $$\hat{n} = \frac{p_1 - p_2}{|p_1 - p_2|}$$

N.B. SOMEWHAT UNSTABLE! (see later)
**CLOSEST POINTS DETERMINATION**

**V-E Collision**

- Project $V$ onto edge $E$

  - Say $E$ defined as start point $o$ and direction $u$

  $$p_E = o + ((v - o) \cdot \hat{u})\hat{u}$$

  - If projection is outside the endpoints, then take the closest endpoint of $E$

  *Standard way of representing a “ray”*
CLOSEST POINTS DETERMINATION

**V-F Collision**

- Project V onto face F

- Say F is defined as normal \( \hat{n} \) and arbitrary point \( f \) on face, then
  \[
  p_F = v - ((v - f) \cdot \hat{n})\hat{n}
  \]

- If projection is outside the polygon, then use V-E case to find closest point
CLOSEST POINTS DETERMINATION

E-E Collision

- For \( E_1 \) defined as start point \( o_1 \) and direction \( u_1 \)
- and \( E_2 \) defined as start point \( o_2 \) and direction \( u_2 \)

\[
p_{E_1} = o_1 + \left( \frac{(o_2-o_1) \cdot (\hat{u}_1 - k \hat{u}_2)}{1-k^2} \right) \hat{u}_1 \quad \text{where} \quad k = \hat{u}_1 \cdot \hat{u}_2
\]

\( p_{E_2} \) is determined as in V-E case

\[
p_{E_2} = \text{closest_point}(p_{E_1} - E_2)
\]
ASIDE: PROOF OF E-E CASE

Assume we know closest point $p_{E_1}$ and $p_{E_2}$.
Then they would have to satisfy

$$p_{E_1} = o_1 + \left( (p_{E_2} - o_1) \cdot \hat{u}_1 \right) \hat{u}_1$$
$$p_{E_2} = o_2 + \left( (p_{E_1} - o_2) \cdot \hat{u}_2 \right) \hat{u}_2$$

Substituting gives

$$p_{E_1} = o_1 + \left( (o_2 + ((p_{E_1} - o_2) \cdot \hat{u}_2) \hat{u}_2 - o_1) \cdot \hat{u}_1 \right) \hat{u}_1$$

Or

$$p_{E_1} = o_1 + \left( \frac{(o_2 - o_1) \cdot (\hat{u}_1 - k \hat{u}_2)}{1 - k^2} \right) \hat{u}_1$$

where $k = u_1 \cdot u_2$
CLOSEST POINTS DETERMINATION

Remaining cases can be solved by decomposing into a set of the previously defined cases

E-F Collision:
- If \( O \) and \( D \) are endpoints of \( E \) and boundary edges of \( F \) are \{ \( E_0, E_1, E_2 \ldots \) \}
- Then \( (E - F) \to \min\{ (O - F), (D - F), (E - E_0), (E - E_1), \ldots \} \)

F-F Collision:
- If boundary edges of \( F_1 \) are \{ \( E_0^1, E_1^1, E_2^1 \ldots \) \} and \( F_2 \) are \{ \( E_0^2, E_1^2, E_2^2 \ldots \) \}
- Then \( (F_1 - F_2) \to \min\{ (E_0^1 - F_2), (E_1^1 - F_2), \ldots, (E_0^2 - F_1), (E_1^2 - F_1), \ldots \} \)

In fact if we assume only closest features are examined, only three cases are really required: \( (V - E), (E - E), (V - F) \)
**CLOSEST POINT ON TRIANGLE**

\( \text{ClosestPointOnTriangleToPoint()} \)

- Finds point on triangle closest to a given point
Separate cases based on which feature Voronoi region point lies in.

A triangle has 7 feature Voronoi regions: 3 vertex regions, 3 edge regions, and 1 face region.
If the point lies in a vertex region, the closest point on the triangle is that vertex. The vertex regions are all tested in a similar manner.

\[ \mathbf{AX} \cdot \mathbf{AB} \leq 0 \]
\[ \mathbf{AX} \cdot \mathbf{AC} \leq 0 \]
If the point lies in an edge region, the closest point on the triangle is the orthogonal projection of the point onto the edge. If neither vertex nor edge regions contain the point, the closest point must lie inside the triangle (in the face region).

\[(BC \times BA) \times BA \cdot BX \geq 0\]
\[AX \cdot AB \geq 0\]
\[BX \cdot BA \geq 0\]
Example in previous slides for Voronoi Region check is for 2D

- should be sufficient for a single triangle intersection in 3D

To extend to 3d polyhedra, a few additional checks for each case are required. See:

- “Real-time Collision Detection” - Christer Ericson. Sec 9.5.2
- “The GJK Algorithm” - Siggraph 2004 Notes by Christer Ericson. Sec 3.1
  - [http://realtimecollisiondetection.net/pubs/SIGGRAPH04_Ericson_GJK_notes.pdf](http://realtimecollisiondetection.net/pubs/SIGGRAPH04_Ericson_GJK_notes.pdf)

“Distance” in WildMagic SDK by David Eberly:

- [http://www.geometrictools.com/LibMathematics/Distance/Distance.html](http://www.geometrictools.com/LibMathematics/Distance/Distance.html)
CONTACT MANIFOLD

Geometry describing the intersection of two objects

For convex polyhedra, this is a polygonal area consisting of:

- Single points
- Line segments
- Closed polygons (possibly with discontinuities/holes if non-convex)
CONTACT FORMATION

We should be able to represent the contact region by the vertices of the intersecting polygonal regions.

- This can be determined from the closest points on the corresponding features.
In real-life we expect a common contact plane and normal

This is not always the case with polyhedral objects

- normal is not defined on an infinitesimal point

Solution:

- Choose normal of feature with higher dimension or if the same then pick a random one of the two
- Take a line between closest points
- Average

Possible stability issues
APPLYING THE CONTACT MODEL

- Now that we have a contact region we are ready for collision response.
- For simple cases, the net or component forces can be calculated based on the contact manifold.
CONCAVE OBJECT CONTACT REGIONS

- Objects with concavity and holes can create multiple disjoint contact regions
- Usually not robust to apply net force in these cases
MUTLIPLE CONTACTS

For multiple simultaneous collisions the problem is also trickier

Some possibilities:

- Use net force (possibly ok for convex-only collisions)
- Use earliest impact – sequential contacts

Correct solutions:

- Integrate force over collision manifold (still not conclusive for disjointed regions)
- Solve simultaneous contacts (e.g. LCP) best method (see later lectures)
SOME PRACTICAL EXAMPLES
EXPANDING POLYTOPE ALGORITHM (EPA)

Recall that the distance from the Minkowski Difference to the Origin is the distance between two polytopes

- For penetrating objects this distance is actually the penetration depth and the direction is the minimum translational distance i.e. the best direction to move them to separate them
- However GJK is primarily intended to tell us whether there is a collision or not (i.e. if a simplex subset of the MD can be found that contains the origin)
- EPA works by expanding this simplex polytope until the closest point is found on the surface of the Configuration Space Object (CSO) to the origin
EPA ALGORITHM (OVERVIEW)

1. Given Configuration Space Object (CSO) and a polytope that contains the origin

2. Find the feature on polytope closest to origin,

3. If this feature lies on the surface of the CSO we are finished

4. Else
   - expand outwards from this feature (usually in outward facing normal of the feature) until a point on the CSO is found - using support function
   - Remove feature from the polytope and replace with features containing the point
   - Repeat from 2,
**Sphere tree Example: Given a BVH node intersection**

- Approximate contact normal:
  
  Line between centres of colliding spheres
  
  \[
  \hat{n} = \frac{c_2 - c_1}{|c_2 - c_1|}
  \]

- Approximate contact point:
  
  Affine translation from centre along contact normal by the ratio of the two spheres
  
  \[
  p = c_1 + (c_2 - c_1) \left( \frac{r_1}{r_1 + r_2} \right)
  \]
Contact Model of boxes is slightly more difficult e.g.

- approximate as a sphere (centre and “radius”); not great for non-uniform boxes
- clipping (N.B. needs to be more efficient than “accurate” narrow phase)

Must be careful of tunnelling

- If linear velocity of a node is greater than the radius of the node, model is not stable...
  - look at parent nodes only (or do primitive test)
  - or look at space-time bounds
OTHER OPTIMIZATION TECHNIQUES
CONTACT LEVELS OF DETAIL

Similar to Multiresolution for display

Simplification must preserve local convexity
CONTACT LEVELS OF DETAIL

Otaduy & Lin Siggraph 2003

For haptic rendering
- requirement of >kHz framerates

Hierarchical multiresolution based on filtered edge collapse

perception of geometric features depends on ratio between contact area and feature - not absolute size of the feature
VOLUMETRIC MODELS & BVH
VOLUMETRIC MODELS

Some Problems (particularly in non-regular BVH)

Solution: cache normals on BV node related to the underlying geometry
CONTACT MANIFOLDS

Pauly et al: Quasi-rigid objects in contact, SCA 2004

Contact model is essentially Inside-outside test with Point cloud representation of object

Also store degree of penetration -> deformation requires some measure of this
CONTACT MANIFOLDS

Galoppo et al, SCA 2006: Dynamic Deformation Textures

GPU extraction of contact manifold for deformable objects
Set up a potentially colliding set (PCS) of objects or sub-objects \( S \).

Object \( O \) is are removed from PCS if it is not fully visible with respect to \( S \).

http://www.randygaul.net/2013/03/28/custom-physics-engine-part-2-manifold-generation/

http://www.codezealot.org/archives/394 [contact points using clipping]

Contact Manifold Generation

http://www.cs.qub.ac.uk/~P.Hanna/CSC3049/default.htm

**OTHER PRACTICAL REFERENCES**

**Contact points using clipping:**
- [http://www.codezealot.org/archives/394](http://www.codezealot.org/archives/394)

**Contact Manifold Generation**
ASSIGNMENT 4&5: NARROW-PHASE, CONTACT MODELLING AND COLLISION RESPONSE

Combined Deadline 06/03/2015
SUMMARY

This 2-week lab is worth ~20% of the full module

Demo due: 22nd February, 2016 at 3-4pm.

There will be a 20% penalty for each day after this

- If late, the onus is on you to email me the code when you have finished and then we will arrange alternate time for demo. The date you email me the code will be taken as date of submission for purposes of the above penalty. It is expected that you will not make further modifications to the demo after this date.

Also submit as part of cumulative submission on 6th February:

- Upto 1 page report on Narrow-phase/Contact Modelling
- Upto 1 page report on Collision Response
- Youtube video of your demo - can be a combined video
- Source code
ASSIGNMENT 4: NARROW-PHASE

Implement Narrow Phase Collision Detection and Contact Modelling

- The minimum requirement is for a convex polyhedron colliding with a plane

Requirements: Given a pair of potentially colliding objects (e.g. from broad-phase)...

- Perform narrow phase intersection testing [you must flag objects that are colliding]
- If objects are colliding, return a contact model for the collision response phase [you must be able to visualise the contact point and direction]

The following solutions are suggested

- BVH of spheres
- Brute force primitive testing
- You may try collisions between two polyhedral objects
- You may try others e.g. SAT, GJK+EPA, Voronoi regions. But please note they could take significant additional time to implement
ASSIGNMENT 5: COLLISION RESPONSE

Implement rigid body collision response for convex polyhedra

- The minimum requirement is for a simple convex body (e.g. cube, prism, cone etc.) colliding with an infinite plane

Requirements: given colliding object(s) and a contact model...

- Apply impulsive force and torque to generate an accurate “bounce”
- Ensure non-interpenetration after collision is resolved

You are not required to implement friction or resting contact
COMBINED MARKING SCHEME

Core marks:
• Narrow Phase Intersection Testing + flagging colliding object(s) [20%]
• Contact modelling + visualising contact model [20%]
• Collision response from [Baraff et al] or [Hecker] implemented [20%]
• Collision response from [Baraff et al] or [Hecker] working [20%]

Extra points [20%] will be given for*:
• Generality of solution e.g. supports different (not hardcoded) polyhedra
• Implementing collisions other than with a plane
• Implementing a working solution for more than 100 objects
• Quality of demo: able to visualise different states, parameters, able to easily change settings, visually interesting
• Implementing GJK + EPA, SAT + Clipping
• Implementing backtracking or conservative advancement

* Please note in some cases this is not a simple add-on (e.g. for GJK+EPA you may have to implement the core components completely differently so this may be overlapped with core and contribute more than just 20%)
IMPLEMENTATION NOTES

For more details see lecture slides
Define plane as any point \( f \) (on plane) and plane normal \( \hat{n} \)

- Then the distance from plane to point \( v \) is given by:

\[
d = \left( (v - f) \cdot \hat{n} \right)
\]
As before, define plane as any point \( f \) (on plane) and plane normal \( \hat{n} \)

- Then the Closest point \( p_F \) on the plane to point \( v \) is given by projecting \( v \) onto the plane:

\[
p_F = v - ((v - f) \cdot \hat{n}) \hat{n}
\]
3. MULTIPLE CONTACTS

For multiple vertices, check all points for collision

- If any are intersecting \((d < 0)\)
  - then collision has occurred
    - EITHER Take closest point \(\min (d_i)\) as point of collision
    - OR find average of colliding points

Optional: backtrack to collision time or move object back by \(d\) along
- \(\hat{n}\) (minimum translational distance)
  - N.B. this is an approximation if object is rotating or not moving linearly!

Should work for convex object with infinite plane BUT for more general collisions this may not be robust
4. VISUALISATION

You MUST demonstrate that everything is working by visualising (where applicable):

- Whether object is intersecting: highlight object in different colour
- Correct closest point is being found: draw line from test point to closest point
- [optional] Correct voronoi region/feature is being found: highlight the relevant feature in whose region the test point lies
- etc…
Collision Response will be discussed in the next class with further implementation notes provided for the second half (Assignment 5) of this two-week assignment.

For anyone skipping ahead, this is largely based on:

- [Hecker] Chris Hecker has a few Gamasutra Articles on the topic that are relatively easy reading:
  - In particular, “Physics Part 4 – The 3rd Dimension”
  - Available http://chrishecker.com/Rigid_Body_Dynamics
  - Rigid Body Dynamics Notes - Collision and Contact
implement a narrow phase collision detection (i.e. primitive intersection tests) between a convex polyhedral object and a plane

- If this is successful, extended to contacts between multiple polyhedral objects

**Compulsory system requirements [worth 60%]**

- Basic contact modelling:
  - MUST return a **boolean response** for the existence of a collision between two objects (one of which might be a plane)
  - MUST return a **contact model** comprising of a collision point and collision normal on each moving object.
  - Demo **must visualise** the relevant contact features and flag collisions

- Basic optimisation:
  - Closest feature determination for a triangle (N.B. complete Lin Canny/GJK is not expected; just a check which feature’s voronoi region a target point is in)
  - N.B. you can show this in a separate demo (not necessarily in the simulated scene)

**Other Improvements may be include further optimisations, generalisation, interesting demo (e.g. multiple objects), more detailed visualisation [worth 40%]**
4. CLOSEST FEATURE DETERMINATION

Instead of brute force,

- check the voronoi regions of features to determine the closest feature of a polygon to a test point \( x \). Start with smallest dimensionality (i.e. Verts then Edges, Polygons)

- Once this is determined, find the closest point \( p_c \) to point \( x \) on the respective feature

N.B. Since the bounded voronoi region has been checked, for edge/plane, this is simply projection of \( p \) onto an infinite plane or line i.e. it eliminates special cases thus, faster than intersecting with line segment or polygon.

MARKING GUIDELINES

Basic Components (~20%)
- Plane and Polyhedron
- Visualisation and Bookkeeping
- Flag object when colliding
- Show Contact points
- Show Contact normals

Collision Detection and Contact Modelling (~40%)
- Boolean test: Polyhedron vs. Plane (based on Brute force point-plane intersection tests) (~10%)
- Closest Point(s) + Collision Normal(s) of Polyhedron to Plane (~15%)
  - Point to Polygon (~10%)
  - Point to Polyhedron (~15%)
  - Polyhedron to Polyhedron (~20%)

Optimisation + options (~40%)
- Voronoi region check Point to Triangle: can be demoed separately (20%)
- Point vs 3D Simplex (~30%)
- Simplex vs. Simplex (~40%)
- Lin Canny feature tracking (~40%)
- Backtracking / minimum-translational distance correction (~15%)
- Generalisation, interesting demo, more detailed visualisation

Bold points are compulsory. Students should attempt some optional elements shown in blue. Note optional elements are not exhaustively listed. Indicative marks for individual optional components are not all cumulative (as some bullet points represent overlapped effort)