Collision detection

Alan Liu
aliu@simcen.usuhs.mil

Surgical Simulation Laboratory
National Capital Area Medical Simulation Center
Uniformed Services University of the Health Sciences
http://simcen.usuhs.mil/miccai2001

Overview

- Motivation
- Classification
- Query type
- Algorithms
- Examples
- Research directions

Motivation

- Need to identify interaction between
  - Tool and tissue
  - Tissue and tissue
  - Tool and tool
- Interactions include
  - Contact
  - Penetration
  - Proximity

A simple collision detection algorithm

- for object_1 in {all objects}
  - for object_2 in {all objects}
    - » if collide(object_1,object_2)
      - add <object_1,object_2> to list
- Algorithm assumes
  - static environment
  - You actually want to detect collision

Collision detection in a dynamic environment

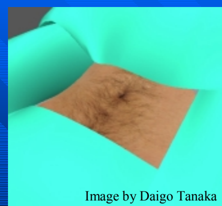
- Two broad classes
  - Detects collisions in continuous spatial-temporal domain
    - » Infinite spatial resolution
  - Detects collisions at discrete intervals in temporal domain
    - » Sampling must ensure collisions are not missed

Query types

- Intersection
  - Collide or not
  - Collision loci
  - Polygon(s) colliding
- Proximity
  - Finding the closest points/polygons
- Prediction
  - Time to collide
  - Priority list

Brute force collision detection

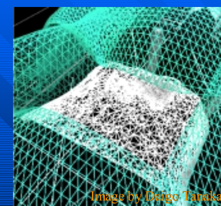
- Check for collisions between objects
  - Self collision
  - Tool-tissue collision
  - tissue-tissue collision
- Too much, need to prune list of objects/polygons to test



Medical Simulation for Surgical Training – MICCAI 2001

Brute force collision detection

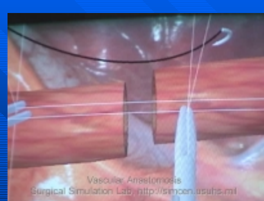
- Check for collisions between objects
  - Self collision
  - Tool-tissue collision
  - tissue-tissue collision
- Too much, need to prune list of objects/polygons to test



Medical Simulation for Surgical Training – MICCAI 2001

Collision detection

- Needs to be done often
  - Every time tools/organs/tissues change position or deform
  - Must be fast
    - » Both visual and haptic rendering
- Types of collisions
  - Tool/tool
  - Tool/tissue
  - Tissue/tissue



Medical Simulation for Surgical Training – MICCAI 2001

Detecting collision between polyhedra

- Feature-based algorithms
  - Features can be vertices, lines, or faces.
- Iteratively converges to the closest feature pair on polyhedra being tested.
- Examples
  - Lin-Canny algorithm
    - » Computes closest features
    - » Linear in number of vertices
    - » Near-constant time if temporal coherence is used
  - V-Clip
    - » Permits penetrating objects
    - » More robust

Medical Simulation for Surgical Training – MICCAI 2001

Detecting collision between polyhedra

- Gilbert, Johnson and Kerrthi Algorithm
  - GJK and enhanced GJK
- Based on techniques in linear programming
- Estimates of penetrating distance possible
  - Uses temporal coherence to run in near constant time

Medical Simulation for Surgical Training – MICCAI 2001

Bounding volumes

- Not cost effective to track distant objects
- Many collision detection libraries use a two-level approach to detecting collisions.
  - Prune uninteresting objects
    - » Too far away
    - » No colliding for next "n" time steps
  - Check for collision between likely objects only
    - » Check between polytopes
    - » Check specific polygons between objects.

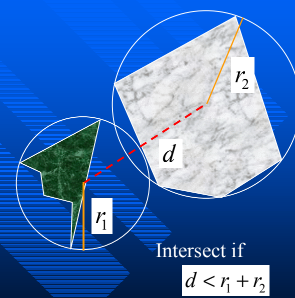
Medical Simulation for Surgical Training – MICCAI 2001

Popular pruning methods

- Bounding volumes
  - Spheres
  - Boxes
    - » Static partition
    - » Axis aligned bounding boxes
    - » Separating Axis Theorem
- Hierarchy of volumes
  - Octrees
  - Sphere trees
  - OBBTrees

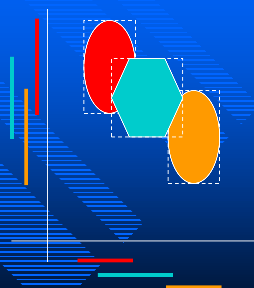
Spheres

- Bound objects by spheres
- Fast, easy to compute
- Can be too conservative
  - Too many false positives if object is long and thin/flat
  - But see [HUBBARD93] and [HUBBARD96]



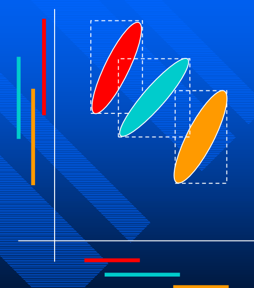
Axis Aligned Bounding Boxes (AABB)

- I-Collide algorithm [COHEN95]
  - Boxes intersect if and only if projections to coordinate axes intersect
  - Uses temporal coherence
    - » Detects all possible intersections in expected linear time



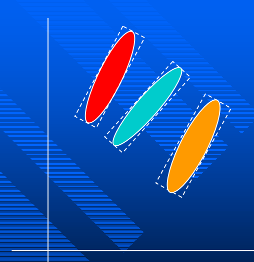
Axis Aligned Bounding Boxes (AABB)

- Inefficient for long thin objects at arbitrary orientations
- Boxes need to be continually resized for
  - rotating objects
  - deforming objects



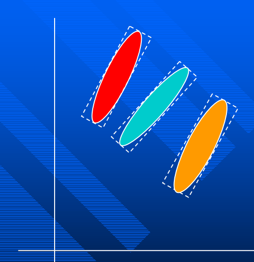
Oriented bounding boxes

- Oriented bounding boxes
- Smallest box that bounds object, not necessarily aligned to coordinate axes
- Separating Axis Theorem [GOTTSCHALK96].
- Two polytopes are disjoint iff.
  - there exists an axis orthogonal to an edge in each polytope, or there exists an axis orthogonal to a face on one polytope



Oriented bounding boxes

- For boxes, need to test at most 15 axes
  - Each box has 3 unique face and 3 unique edge orientations
  - Thus, 3 faces (on one box) + 3 faces (on the other box) + 9 edge combinations



Static partition

- Partition space into fixed sized boxes
- Build hash table mapping space to polygons
- Advantages
  - Fast (expected constant time)
  - Deformations easily handled
- Disadvantages
  - Memory overhead

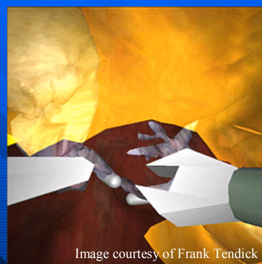


Image courtesy of Frank Tendick

Hierarchy of bounding volumes

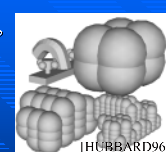
- Need tighter fit with object
- Collision detection reduced to tree search
  - Typically O(lg N) time
- Types of bounding volume hierarchies
  - Octrees [SAMMET98]
  - Sphere trees [HUBBARD93]
  - AABB Trees [BERGEN97]
  - OBB Trees [GOTTSCHALK96]

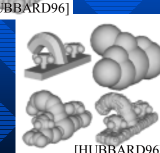
Octrees
[SAMMET98]

- Start with AABB encompassing entire object
  - Recursively subdivide box into four sub-boxes until a lower limit is reached
  - Relatively easy to construct
  - Updates are non-trivial
    - » Movement
    - » Deformation

Sphere trees
[HUBBARD93]

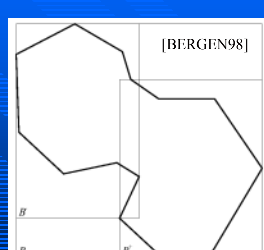
- Instead of using boxes, use spheres
- Methods for building hierarchy
  - Octree subdivision [HUBBARD93]
  - Medial axis based [HUBBARD96]


[HUBBARD96]


[HUBBARD96]

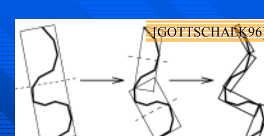
AABB Trees
[BERGEN98]

- Recursively subdivide AABB by partitioning along longest axis
  - Equal number of primitives on either side of partition
- Refit boxes after deformation
  - AABB of child AABBs is smallest AABB enclosing all primitives.
- Refitted boxes have more overlap than rebuilt boxes


[BERGEN98]

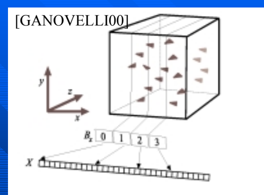
OBBTrees
[GOTTSCHALK96]

- Recursively subdivide OBBs
  - Divide along longest/2nd longest/shortest axis
  - Repeat division until not possible to divide on any axis
  - Use Separating Axis Theorem to determine intersection


[GOTTSCHALK96]

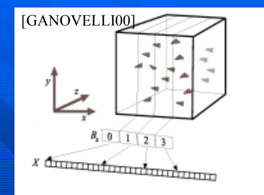
Bucket trees
[GANOVELLI00]

- For each axis, maintain sorted list of polygon coordinates
- Use octree to partition space
- Leaves point to polygon with lowest coordinate value in each array
  - Each leaf contains a "bucket" of polygons



Bucket trees

- When object deforms, polygons move to different bucket
  - Change pointer at leaves



Sphere trees	AABB Tree	OBB Tree	Bucket Trees
• Rotationally invariant • Intersection test is very simple • Approximation to underlying object may be poor	• No need to rebuilt tree for deformable objects • Easy to build • Relatively easy to test for intersections • Needs more intersection tests compared to OBB Tree (object dependent)	• Tight fit with underlying object • Faster • Relatively easy to test for intersections • Needs fewer intersection tests compared to AABB Tree (object dependent)	• Updating tree for deformable objects requires little effort • Tree can get unbalanced
Efficient for rejection tests, less efficient for cases where multiple intersections exists			

Hardware based collision detection
[LOMBARDO99]

- Dynamic collision detection
  - No need to take discrete samples in time
- No additional data structures
- Uses commonly available graphics hardware
  - OpenGL based
- Detects multiple contact points in one pass
- Runs at interactive frame rates
  - Depending on hardware

Hardware based collision detection
[LOMBARDO99]

- Laparoscopic instrument moves about a pivot point
- At each time step, instrument moves in (nearly) a straight line.
- Volume swept out by instrument approximates a viewing frustum
- Use OpenGL define that frustum, then render
- Anything visible must be due to objects intersecting path of instrument

Hardware based collision detection
[LOMBARDO99]

Using the Author's method:

processor graphic	R10000 195 MHz Onyx2 IR	DEC alpha 500 MHz 4D60	Pentium2 333Mhz software (Linux Mesa)	Pentium2 333Mhz 3Dfx Voodoo2 (Linux Mesa)
static dynamic	0.13 ms 0.16 ms	0.09 ms 0.11 ms	2.2 ms 3.0 ms	1.7 ms 2.3 ms

Using the Obb tree method:

processor	R10000 195 MHz	DEC alpha 500 MHz	Pentium2 333Mhz
Precomputations static dynamic	24.1 ms 0.63 ms 0.76 ms	15.7 ms 0.44 ms 0.48 ms	35.6 ms 1.0 ms 1.2 ms

NB: *static* means considering a single position for the tool
dynamic means considering the tool positions during a time interval

Open questions

- Many, many points of contact
  - Organ/organ, Suturing, Palpation
  - Many current algorithms not efficient at solving this problem
- Deformation, change in topology
- Self collision
  - Knot tying
- Collision detection using a volumetric representation
- Need to do everything in real-time

Collision detection code

- Lin-Canny algorithm
  - http://www.cs.berkeley.edu/~mirtich/collDet.html
- I-COLLIDE,V-COLLIDE and many others
  - http://www.cs.unc.edu/~geom/
- SOLID (AABB Trees)
  - http://www.win.tue.nl/~gino/solid/
- Enhanced GJK
  - http://web.comlab.ox.ac.uk/oucl/work/stephen.cameron /distances/index.html

Bibliography

- [BERGEN97] van den Bergen G. "Efficient collision detection of complex deformable models using AABB trees." Journal of Graphics Tools, 2(4):1--13, 1997. pp. 1-13.
- [HUBBARD96] Hubbard, P.M. "Approximating Polyhedra with Spheres for Time-Critical Collision Detection." ACM Trans. on Graphics, 15(3), 1996. pp. 179-210.
- [HUBBARD93] Hubbard, P.M. "Interactive collision detection." Proceedings of the IEEE Symposium on Research Frontiers in Virtual Reality, October 25-26, 1993. pp. 24-31.
- [SAMMET98] Samet H., Webber R.E. "Hierarchical data structures and algorithms for computer graphics, part I: Fundamentals." IEEE Computer Graphics and Applications, vol. 8, no 3., May 1998. pp. 48-68.
- [LIN98] Lin, Ming C., Gottschalk S. "Collision detection between geometric models: a survey" In Proc. of IMA Conference on Mathematics of Surfaces, 1998. ftp://ftp.cs.unc.edu/pub/users/manocha/PAPERS/COLLISION/cms.pdf.

Bibliography

- [GANOVELLI00] Ganovelli F., Dingliana J., O'Sullivan C. "BucketTree: Improving collision detection between deformable objects" Conference on Computer Graphics (SCCG2000), Bratislava, April 2000. pp. 156-163. http://citeseer.nj.nec.com/344463.html.
- [GOTTSCHALK96] Gottschalk S., Lin M.C., Manocha D. "OBBtree: A hierarchical structure for rapid interference detection." SIGGRAPH '96 Proc. 1996. pp. 171-180. http://citeseer.nj.nec.com/43036.html,ftp://cs.unc.edu/pub/users/manocha/PAPERS/COLLISION/obb.ps.gz
- [COHEN95] Cohen J.D., Lin M.C., Manocha D., Ponamgi M.K. "I-COLLIDE: An interactive and exact collision detection system for large-scale environments." Proceedings of the 1995 Symposium on Interactive 3D Graphics (Monterey, California), 1995. pp. 189-196.
- [LOMBARDO99] Lombardo J.C., Gascuel M.P., Neyret F. "Real-time collision detection for virtual surgery." Proceedings of Computer Animation '99, May 1999. pp. 33-39. http://citeseer.nj.nec.com/lombardo99realtime.html