

Fast Collision Detection Between General Polyhedral Models

Ming C. Lin

lin@cs.unc.edu

<http://www.cs.unc.edu/~lin>

<http://gamma.cs.unc.edu/>

University of North Carolina at Chapel
Hill

Ming Lin

Methods for General Models

- Decompose into convex pieces, and take minimum over all pairs of pieces:
 - Optimal (minimal) model decomposition is NP-hard.
 - Approximation algorithms exist for closed solids, but what about a list of triangles?
- Collection of triangles/polygons:
 - $n*m$ pairs of triangles - brute force expensive
 - Hierarchical representations used to accelerate minimum finding

Ming Lin

Hierarchical Representations



- Two Common Types:
 - **Bounding volume hierarchies** – trees of spheres, ellipses, cubes, axis-aligned bounding boxes (AABBs), oriented bounding boxes (OBBs), K-dop, SSV, etc.
 - **Spatial decomposition** - BSP, K-d trees, octrees, MSP tree, R-trees, grids/cells, space-time bounds, etc.
- Do very well in “rejection tests”, when objects are far apart
- Performance may slow down, when the two objects are in close proximity and can have multiple contacts

Ming Lin

BVH vs. Spatial Partitioning

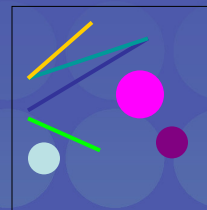
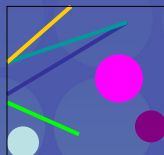


BVH:

- Object centric
- Spatial redundancy

SP:

- Space centric
- Object redundancy



Ming Lin

BVH vs. Spatial Partitioning

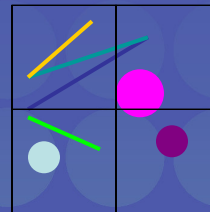
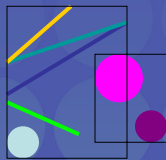


BVH:

- Object centric
- Spatial redundancy

SP:

- Space centric
- Object redundancy



Ming Lin

BVH vs. Spatial Partitioning

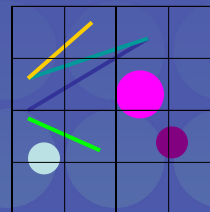
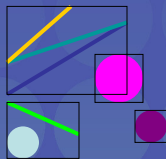


BVH:

- Object centric
- Spatial redundancy

SP:

- Space centric
- Object redundancy



Ming Lin

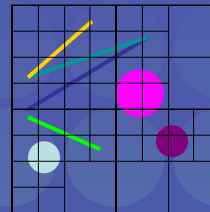
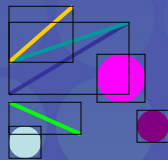
BVH vs. Spatial Partitioning

BVH:

- Object centric
- Spatial redundancy

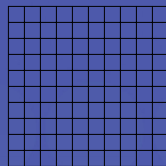
SP:

- Space centric
- Object redundancy

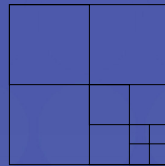


Ming Lin

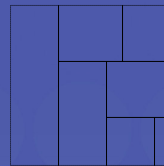
Spatial Data Structures & Subdivision



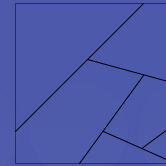
Uniform Spatial Sub



Quadtree/Octree



kd-tree



BSP-tree

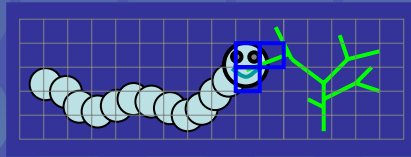
- Many others.....
(see the lecture notes)

Ming Lin

Uniform Spatial Subdivision



- Decompose the objects (the entire simulated environment) into identical cells arranged in a fixed, regular grids (equal size boxes or voxels)
- To represent an object, only need to decide which cells are occupied. To perform collision detection, check if any cell is occupied by two object
- Storage: to represent an object at resolution of n voxels per dimension requires upto n^3 cells
- Accuracy: solids can only be “approximated”



Octrees

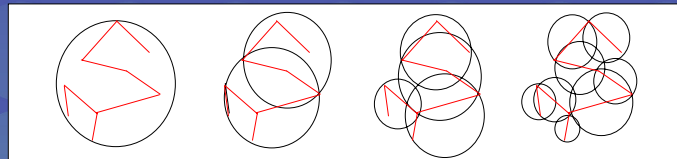


- *Quadtree* is derived by subdividing a 2D-plane in both dimensions to form quadrants
- Octrees are a 3D-extension of quadtree
- Use divide-and-conquer
- Reduce storage requirements (in comparison to grids/voxels)



Bounding Volume Hierarchies

- Model Hierarchy:
 - each node has a simple volume that bounds a set of triangles
 - children contain volumes that each bound a different portion of the parent's triangles
 - The leaves of the hierarchy usually contain individual triangles
- A binary bounding volume hierarchy:



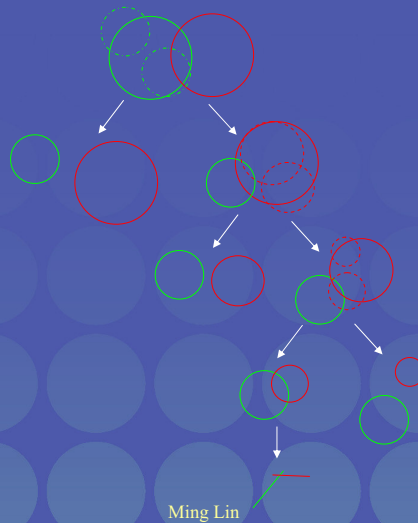
Ming Lin

Type of Bounding Volumes

- Spheres
- Ellipsoids
- Axis-Aligned Bounding Boxes (AABB)
- Oriented Bounding Boxes (OBBs)
- Convex Hulls
- k -Discrete Orientation Polytopes (k -dop)
- Spherical Shells
- Swept-Sphere Volumes (SSVs)
 - Point Swept Spheres (PSS)
 - Line Swept Spheres (LSS)
 - Rectangle Swept Spheres (RSS)
 - Triangle Swept Spheres (TSS)

Ming Lin

BVH-Based Collision Detection



Ming Lin

Collision Detection using BVH



1. Check for collision between two parent nodes (starting from the roots of two given trees)
2. If there is no interference between two parents,
3. Then stop and report "no collision"
4. Else All children of one parent node are checked against all children of the other node
5. If there is a collision between the children
6. Then If at leave nodes
7. Then report "collision"
8. Else go to Step 4
9. Else stop and report "no collision"

Ming Lin

Evaluating Bounding Volume Hierarchies



Cost Function:

$$F = N_u \times C_u + N_{bv} \times C_{bv} + N_p \times C_p$$

F : total cost function for interference detection

N_u : no. of bounding volumes updated

C_u : cost of updating a bounding volume,

N_{bv} : no. of bounding volume pair overlap tests

C_{bv} : cost of overlap test between 2 BVs

N_p : no. of primitive pairs tested for interference

C_p : cost of testing 2 primitives for interference

Ming Lin

Designing Bounding Volume Hierarchies



The choice governed by these constraints:

- It should fit the original model as tightly as possible (to lower N_{bv} and N_p)
- Testing two such volumes for overlap should be as fast as possible (to lower C_{bv})
- It should require the BV updates as infrequently as possible (to lower N_u)

Ming Lin

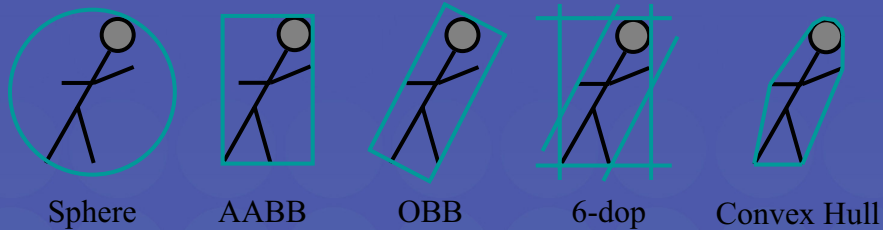
Observations



- Simple primitives (spheres, AABBs, etc.) do very well with respect to the second constraint. But they cannot fit some long skinny primitives tightly.
- More complex primitives (minimal ellipsoids, OBBs, etc.) provide tight fits, but checking for overlap between them is relatively expensive.
- Cost of BV updates needs to be considered.

Ming Lin

Trade-off in Choosing BV's



Sphere

AABB

OBB

6-dop

Convex Hull

increasing complexity & tightness of fit

decreasing cost of (overlap tests + BV update)

Ming Lin

Building Hierarchies



- Choices of Bounding Volumes
 - cost function & constraints
- Top-Down vs. Bottom-up
 - speed vs. fitting
- Depth vs. breadth
 - branching factors
- Splitting factors
 - where & how

Ming Lin

Sphere-Trees



- A *sphere-tree* is a hierarchy of sets of spheres, used to approximate an object
- Advantages:
 - Simplicity in checking overlaps between two bounding spheres
 - Invariant to rotations and can apply the same transformation to the centers, if objects are rigid
- Shortcomings:
 - Not always the best approximation (esp bad for long, skinny objects)
 - Lack of good methods on building sphere-trees

Ming Lin

Methods for Building Sphere-Trees



- "Tile" the triangles and build the tree bottom-up
- Covering each vertex with a sphere and group them together
- Start with an octree and "tweak"
- Compute the medial axis and use it as a skeleton for multi-res sphere-covering
- Others.....

Ming Lin

k-DOP's



- *k*-dop: *k*-discrete orientation polytope a convex polytope whose facets are determined by half-spaces whose outward normals come from a small fixed set of *k* orientations
- For example:
 - In 2D, an 8-dop is determined by the orientation at +/- {45,90,135,180} degrees
 - In 3D, an AABB is a 6-dop with orientation vectors determined by the +/-coordinate axes.

Ming Lin

Choices of k -dops in 3D



- 6-dop: defined by coordinate axes
- 14-dop: defined by the vectors $(1,0,0)$, $(0,1,0)$, $(0,0,1)$, $(1,1,1)$, $(1,-1,1)$, $(1,1,-1)$ and $(1,-1,-1)$
- 18-dop: defined by the vectors $(1,0,0)$, $(0,1,0)$, $(0,0,1)$, $(1,1,0)$, $(1,0,1)$, $(0,1,1)$, $(1,-1,0)$, $(1,0,-1)$ and $(0,1,-1)$
- 26-dop: defined by the vectors $(1,0,0)$, $(0,1,0)$, $(0,0,1)$, $(1,1,1)$, $(1,-1,1)$, $(1,1,-1)$, $(1,-1,-1)$, $(1,1,0)$, $(1,0,1)$, $(0,1,1)$, $(1,-1,0)$, $(1,0,-1)$ and $(0,1,-1)$

Ming Lin

Building Trees of k -dops

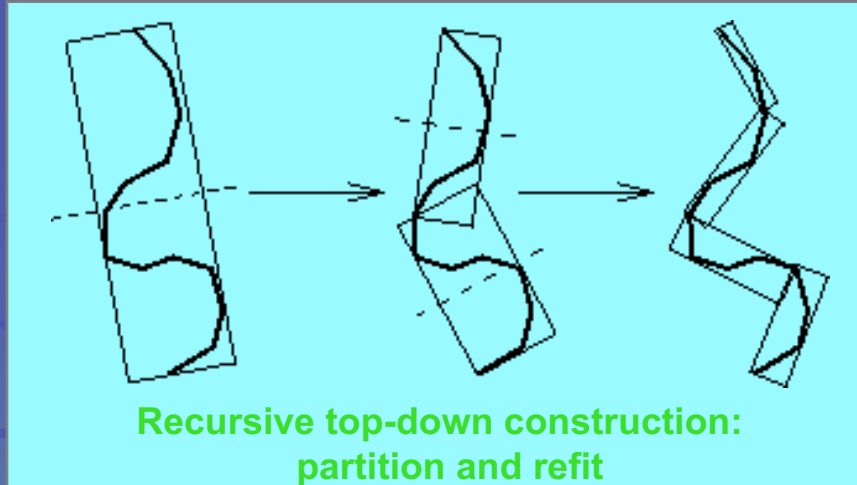


The major issue is updating the k -dops:

- Use Hill Climbing (as proposed in I-Collide) to update the min/max along each $k/2$ directions by comparing with the neighboring vertices
- But, the object may not be convex..... Use the approximation (convex hull vs. another k -dop)

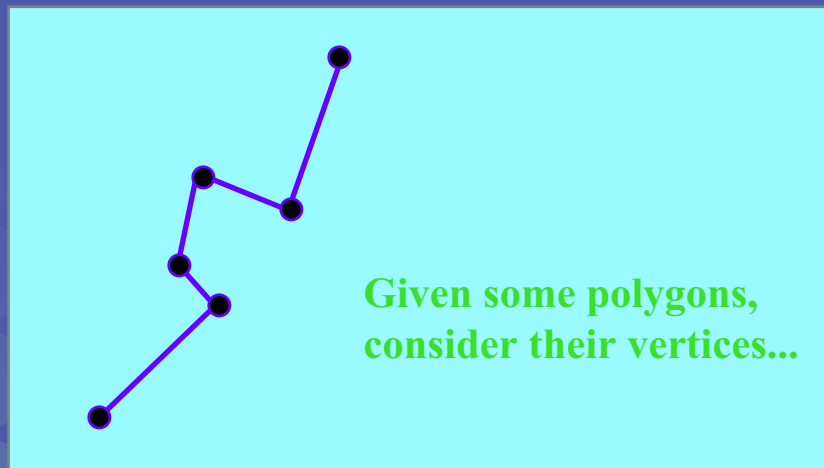
Ming Lin

Building an OBBTree



Ming Lin

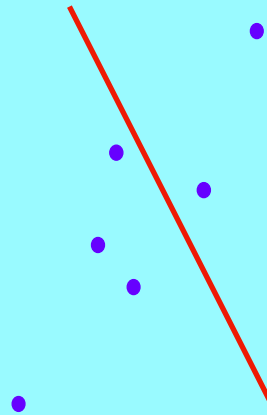
Building an OBB Tree



Ming Lin

Building an OBB Tree

... and an arbitrary line

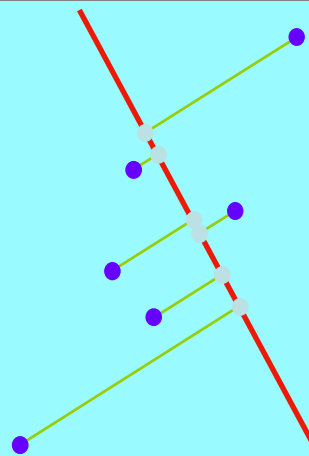


Ming Lin

Building an OBB Tree

Project onto the line

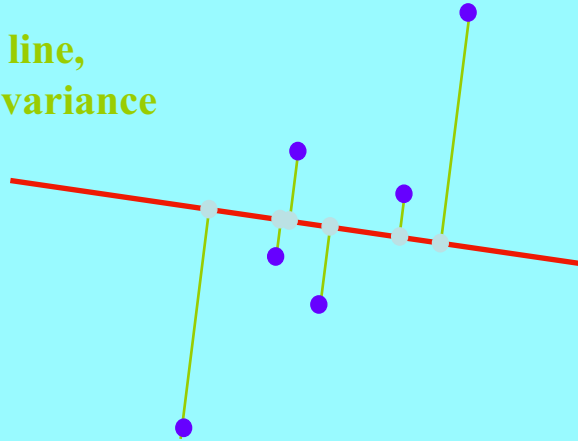
Consider variance of
distribution on the line



Ming Lin

Building an OBB Tree

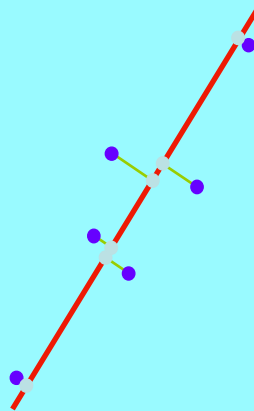
Different line,
different variance



Ming Lin

Building an OBB Tree

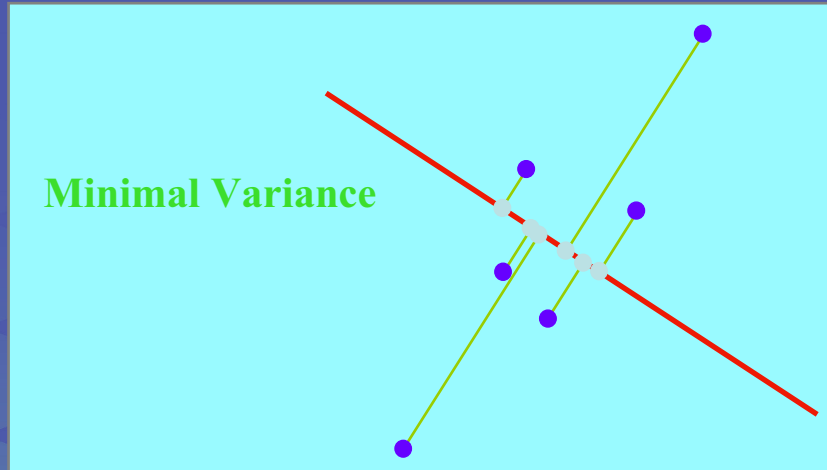
Maximum Variance



Ming Lin

Building an OBB Tree

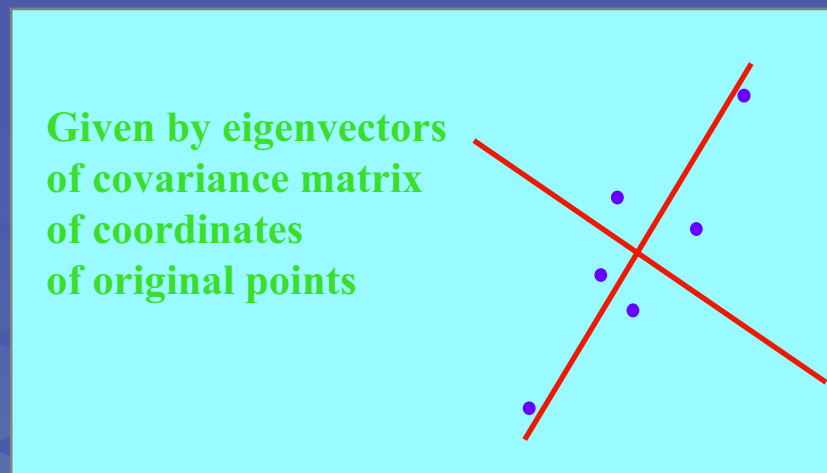
Minimal Variance



Ming Lin

Building an OBB Tree

Given by eigenvectors
of covariance matrix
of coordinates
of original points

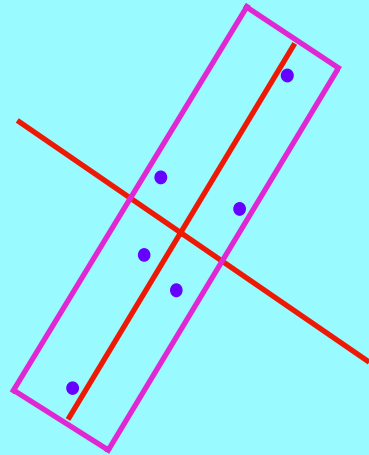


Ming Lin

Building an OBB Tree



Choose bounding box oriented this way



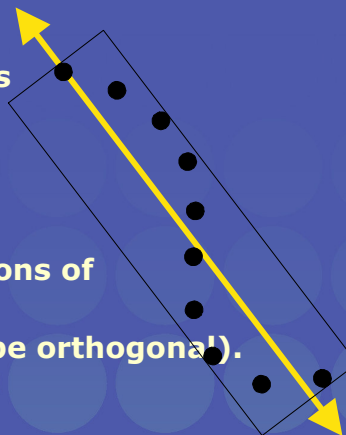
Ming Lin

Building an OBB Tree: Fitting



Covariance matrix of point coordinates describes statistical spread of cloud.

OBB is aligned with directions of greatest and least spread (which are guaranteed to be orthogonal!).



Ming Lin

Fitting OBBs

- Let the vertices of the i 'th triangle be the points a^i, b^i , and c^i , then the mean μ and covariance matrix C can be expressed in vector notation as:

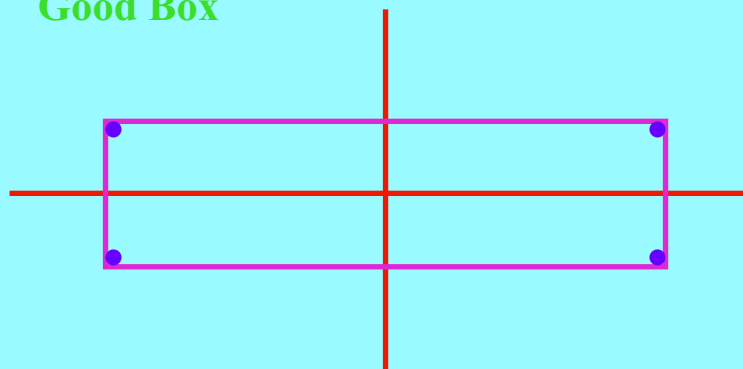
$$\mu = \frac{1}{3n} \sum_{i=0}^n (a^i + b^i + c^i),$$
$$C_{j k} = \frac{1}{3n} \sum_{i=0}^n (\bar{a}_j^i \bar{a}_k^i + \bar{b}_j^i \bar{b}_k^i + \bar{c}_j^i \bar{c}_k^i), \quad 1 \leq j, k \leq 3$$

$$\bar{a}^i = a^i - \mu, \quad \bar{b}^i = b^i - \mu, \quad \bar{c}^i = c^i - \mu.$$

Ming Lin

Building an OBB Tree

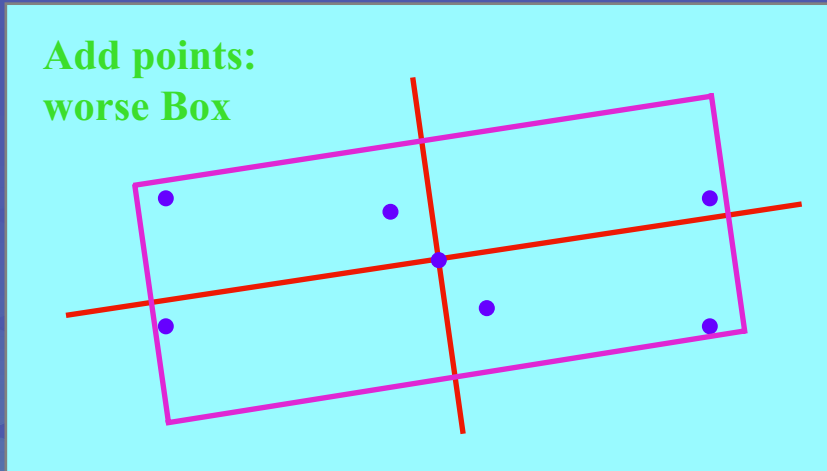
Good Box



Ming Lin

Building an OBB Tree

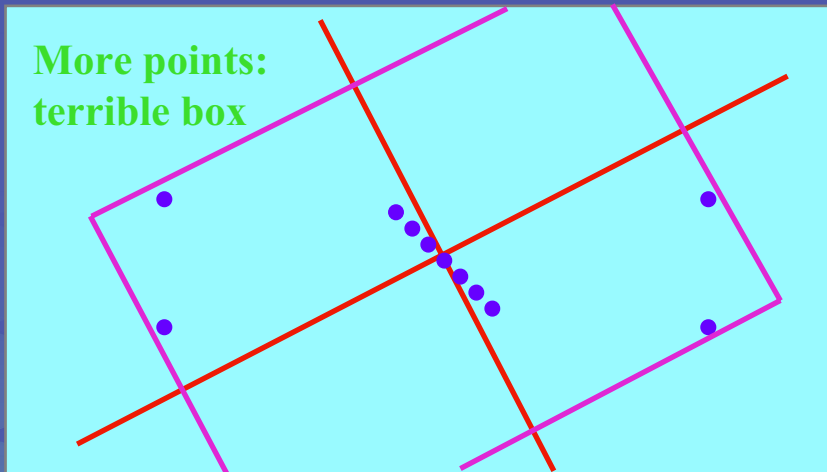
Add points:
worse Box



Ming Lin

Building an OBB Tree

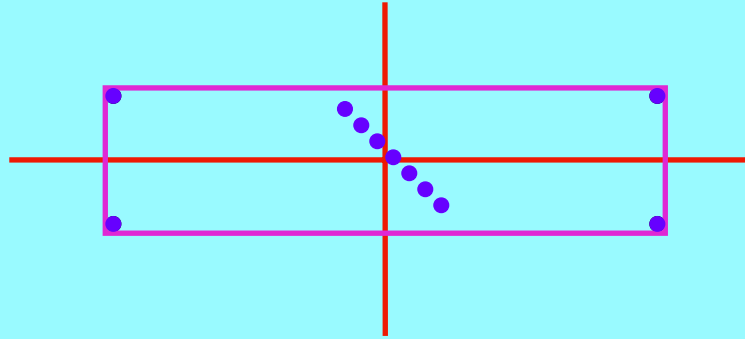
More points:
terrible box



Ming Lin

Building an OBB Tree

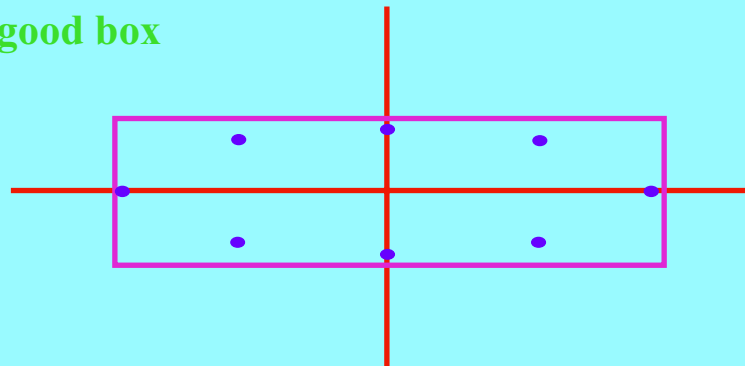
Compute with extremal points only



Ming Lin

Building an OBB Tree

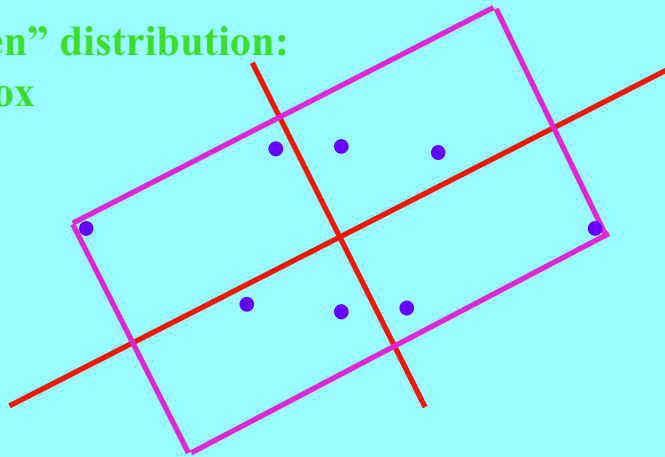
“Even” distribution:
good box



Ming Lin

Building an OBB Tree

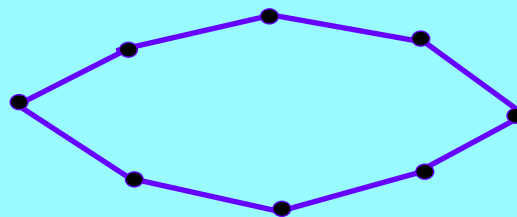
**“Uneven” distribution:
bad box**



Ming Lin

Building an OBB Tree

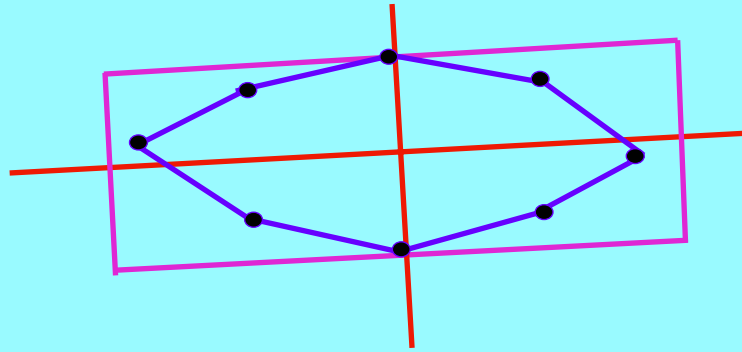
Fix: Compute facets of convex hull...



Ming Lin

Building an OBB Tree

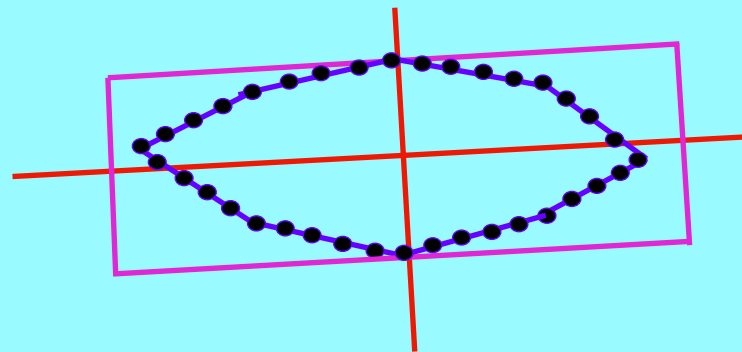
Better: Integrate over facets



Ming Lin

Building an OBB Tree

... and sample them uniformly



Ming Lin

Building an OBB Tree: Summary



OBB Fitting algorithm:

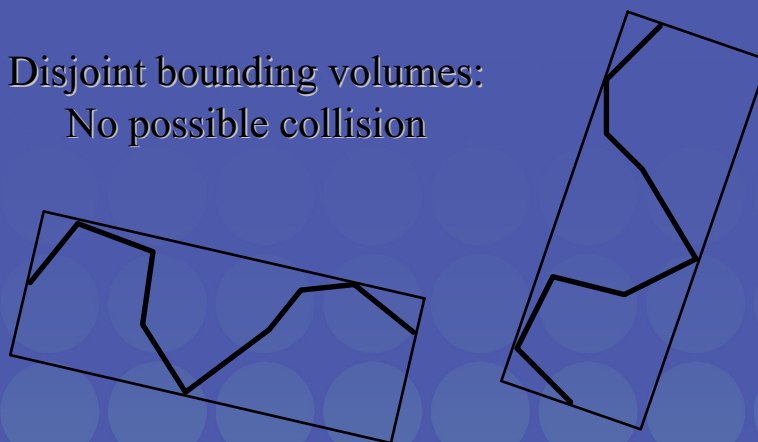
- covariance-based
- use of convex hull
- not foiled by extreme distributions
- $O(n \log n)$ fitting time for single BV
- $O(n \log^2 n)$ fitting time for entire tree

Ming Lin

Tree Traversal



Disjoint bounding volumes:
No possible collision

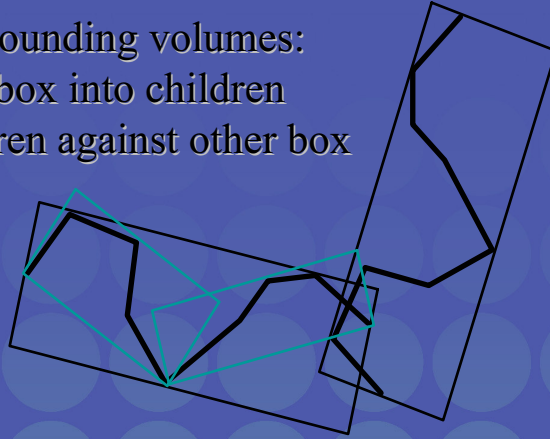


Ming Lin

Tree Traversal

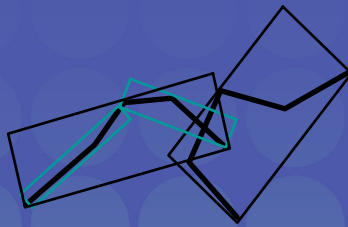
Overlapping bounding volumes:

- split one box into children
- test children against other box



Ming Lin

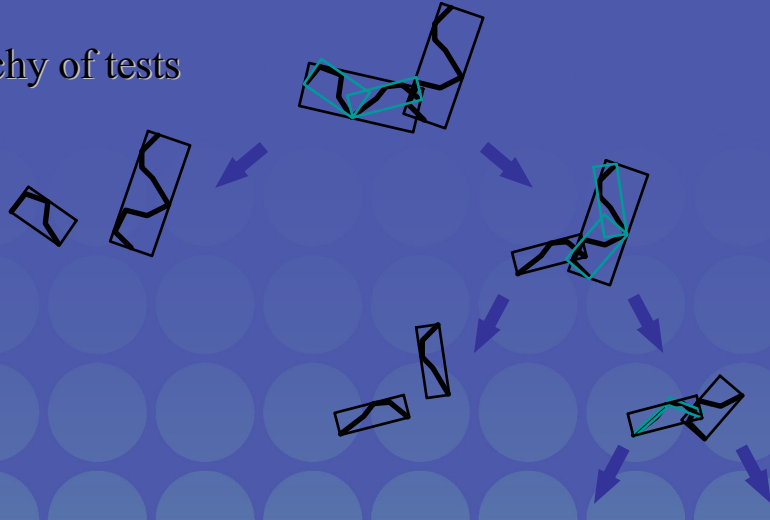
Tree Traversal



Ming Lin

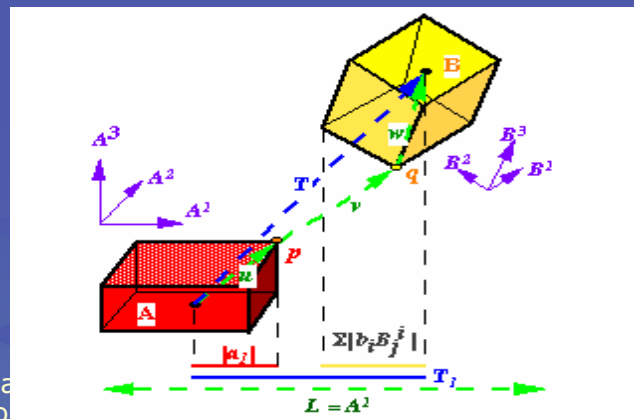
Tree Traversal

Hierarchy of tests



Ming Lin

Separating Axis Theorem



• L is a
disjo

ecome

Ming Lin

Separating Axis Theorem



Two polytopes A and B are disjoint iff there exists a separating axis which is:

perpendicular to a face from either
or
perpendicular to an edge from each

Ming Lin

Implications of Theorem



Given two generic polytopes, each with E edges and F faces, number of candidate axes to test is:

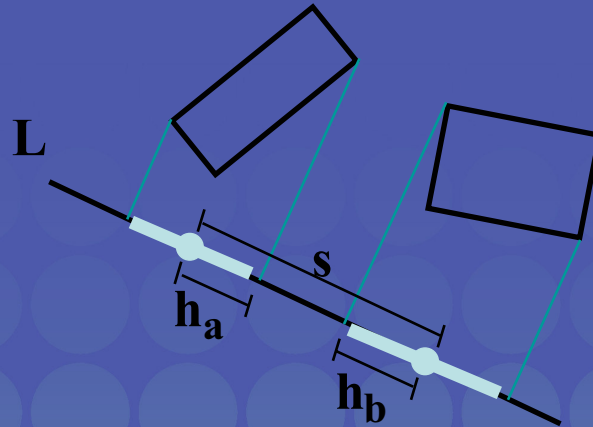
$$2F + E^2$$

OBBs have only $E = 3$ distinct edge directions, and only $F = 3$ distinct face normals. OBBs need at most 15 axis tests.

Because edge directions and normals each form orthogonal frames, the axis tests are rather simple.

Ming Lin

OBB Overlap Test: An Axis Test



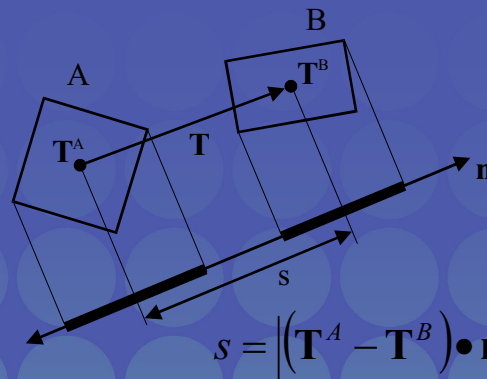
L is a separating axis iff: $s > h_a + h_b$

Ming Lin

OBB Overlap Test: Axis Test Details



Box centers project to interval midpoints, so midpoint separation is length of vector \mathbf{T} 's image.

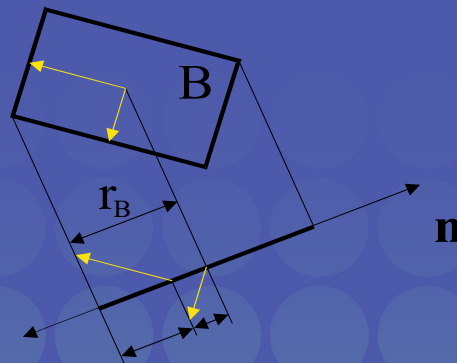


Ming Lin

OBB Overlap Test: Axis Test Details



- Half-length of interval is sum of box axis images.



$$r_B = b_1 |\mathbf{R}_1^B \cdot \mathbf{n}| + b_2 |\mathbf{R}_2^B \cdot \mathbf{n}| + b_3 |\mathbf{R}_3^B \cdot \mathbf{n}|$$

Ming Lin

OBB Overlap Test



- Typical axis test for 3-space.

```
s = fabs(T2 * R11 - T1 * R21);
```

```
ha = a1 * Rf21 + a2 * Rf11;
```

```
hb = b0 * Rf02 + b2 * Rf00;
```

```
if (s > (ha + hb)) return 0;
```

- Up to 15 tests required.

Ming Lin

OBB Overlap Test



- Strengths of this overlap test:
 - 89 to 252 arithmetic operations per box overlap test
 - Simple guard against arithmetic error
 - No special cases for parallel/coincident faces, edges, or vertices
 - No special cases for degenerate boxes
 - No conditioning problems
 - Good candidate for micro-coding

Ming Lin

OBB Overlap Tests: Comparison

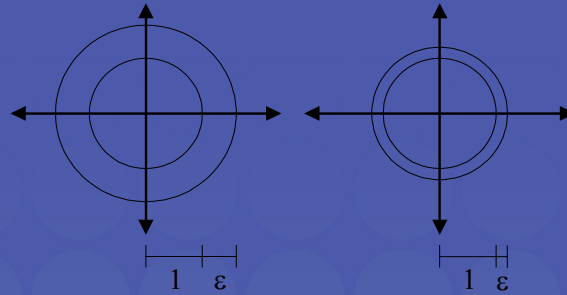


Test Method	Speed(us)
Separating Axis	6.26
GJK	66.30
LP	217.00

Benchmarks performed on SGI Max Impact,
250 MHz MIPS R4400 CPU, MIPS R4000 FPU

Ming Lin

Parallel Close Proximity

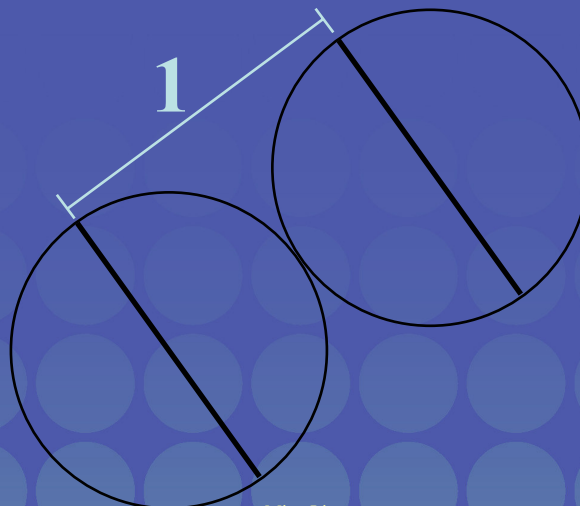


Two models are in *parallel close proximity* when every point on each model is a given fixed distance (ϵ) from the other model.

Q: How does the number of BV tests increase as the gap size decreases?

Ming Lin

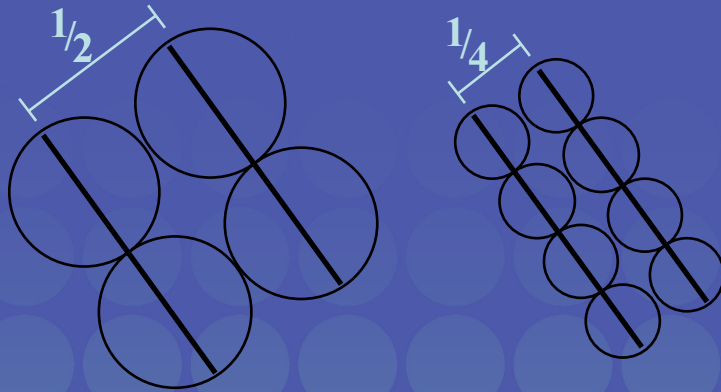
Parallel Close Proximity: Convergence



Ming Lin

Parallel Close Proximity: Convergence

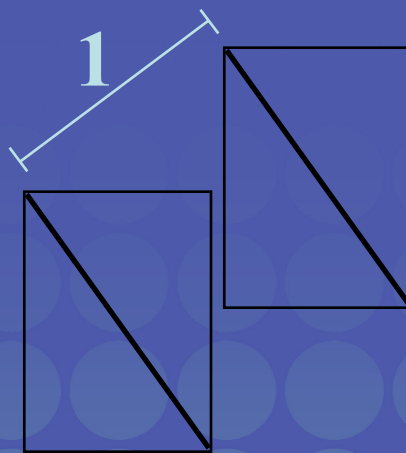
SIGGRAPH2004



Ming Lin

Parallel Close Proximity: Convergence

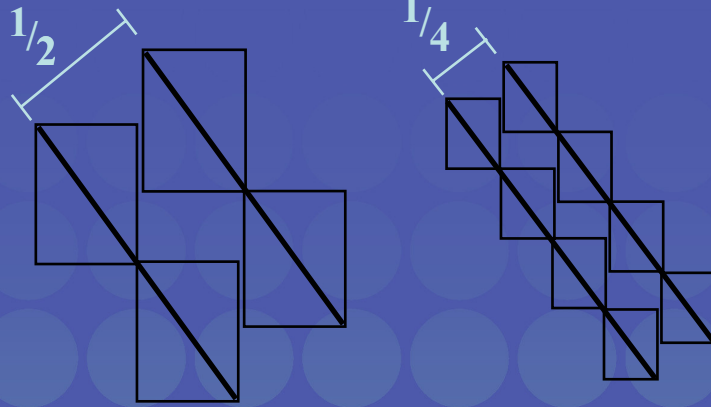
SIGGRAPH2004



Ming Lin

Parallel Close Proximity: Convergence

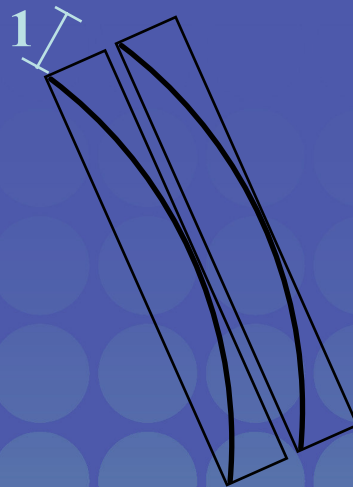
SIGGRAPH2004



Ming Lin

Parallel Close Proximity: Convergence

SIGGRAPH2004

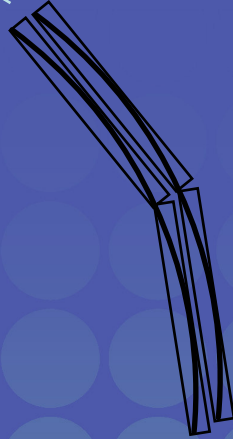


Ming Lin

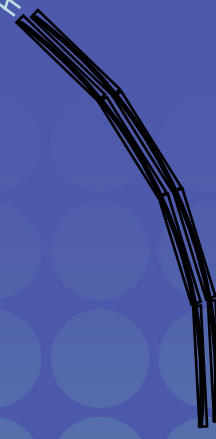
Parallel Close Proximity: Convergence

SIGGRAPH2004

$1/4$ I



$1/16$ I

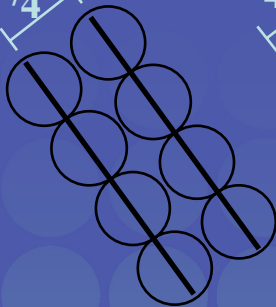


Ming Lin

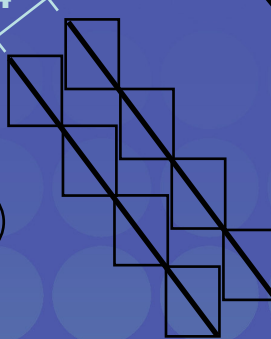
Parallel Close Proximity: Convergence

SIGGRAPH2004

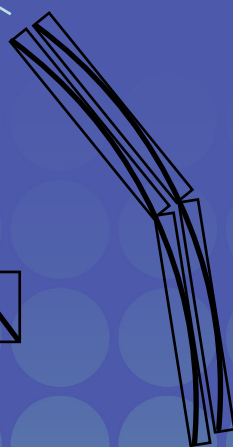
$1/4$ I



$1/4$ I

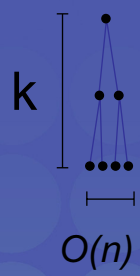


$1/4$ I

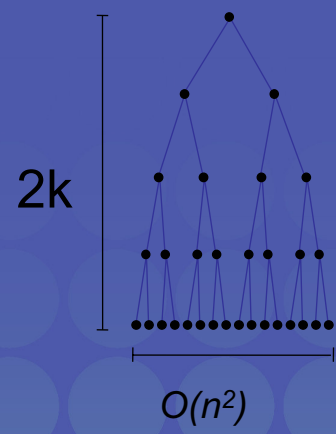


Ming Lin

Performance: Overlap Tests



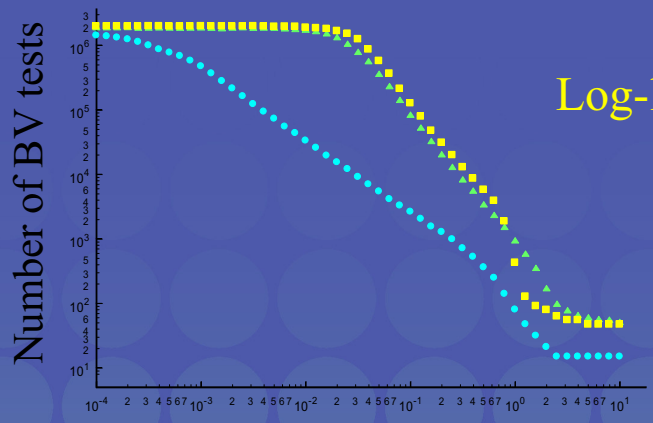
OBBs



Spheres & AABBs

Ming Lin

Parallel Close Proximity: Experiment

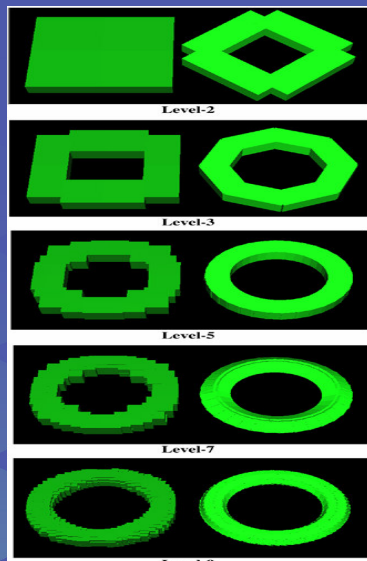


Log-log plot

OBBs asymptotically outperform AABBs and spheres

Ming Lin

Example: AABB's vs. OBB's

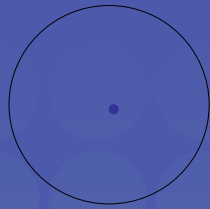


Approximation
of a Torus

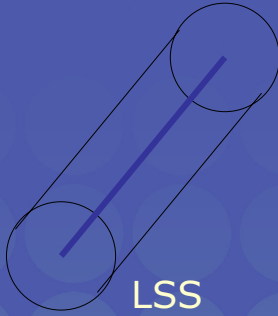
Implementation: RAPID

- Available at:
<http://www.cs.unc.edu/~geom/OBB>
- Part of **V-COLLIDE**:
http://www.cs.unc.edu/~geom/V_COLLIDE
- Thousands of users have ftp'ed the code
- Used for virtual prototyping, dynamic simulation, robotics & computer animation

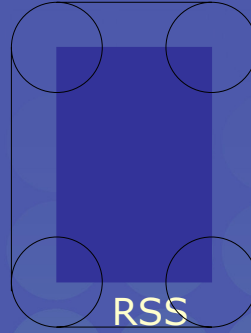
Hybrid Hierarchy of Swept Sphere Volumes



PSS



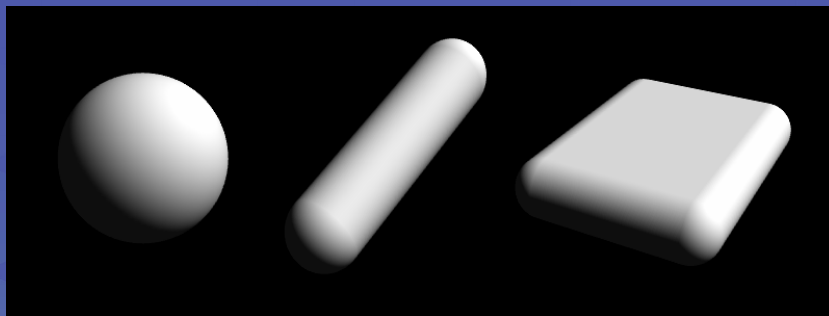
LSS
[LGLM99]



RSS

Ming Lin

Swept Sphere Volumes (S-topes)



PSS

LSS

RSS

Ming Lin

SSV Fitting



- Use OBB's code based upon *Principle Component Analysis*
- For PSS, use the largest dimension as the radius
- For LSS, use the two largest dimensions as the length and radius
- For RSS, use all three dimensions

Ming Lin

Overlap Test



- One routine that can perform overlap tests between all possible combination of CORE primitives of SSV(s).
- The routine is a specialized test based on Voronoi regions and OBB overlap test.
- It is faster than GJK.

Ming Lin

Hybrid BVH's Based on SSVs



- Use a simpler BV when it prunes search equally well - benefit from lower cost of BV overlap tests
- Overlap test (based on Lin-Canny & OBB overlap test) between all pairs of BV's in a BV family is unified
- Complications
 - deciding which BV to use either *dynamically* or *statically*

Ming Lin

PQP: Implementation



- Library written in C++
- Good for any proximity query
- 5-20x speed-up in *distance computation* over prior methods
- Available at
<http://www.cs.unc.edu/~geom/SSV/>

Ming Lin

References



- [OBB-Tree: A Hierarchical Structure for Rapid Interference Detection](#), by S. Gottschalk, M. Lin and D. Manocha, Proc. of ACM Siggraph, 1996.
- [Rapid and Accurate Contact Determination between Spline Models using ShellTrees](#), by S. Krishnan, M. Gopi, M. Lin, D. Manocha and A. Pattekar, Proc. of Eurographics 1998.
- [Fast Proximity Queries with Swept Sphere Volumes](#), by Eric Larsen, Stefan Gottschalk, Ming C. Lin, Dinesh Manocha, Technical report TR99-018, UNC-CH, CS Dept, 1999. (Part of the paper in Proc. of IEEE ICRA'2000)

Fast Proximity Queries Between Convex Polyhedra

Ming C. Lin
lin@cs.unc.edu
<http://www.cs.unc.edu/~lin>
<http://gamma.cs.unc.edu/>
University of North Carolina at Chapel Hill

Ming Lin

Voronoi Diagrams

- Given a set S of n points in R^2 , for each point p_i in S , there is the set of points (x, y) in the plane that are closer to p_i than any other point in S , called Voronoi polygons. The collection of n Voronoi polygons given the n points in the set S is the "Voronoi diagram", $Vor(S)$, of the point set S .

Intuition: To partition the plane into regions, each of these is the set of points that are closer to a point p_i in S than any other. The partition is based on the set of closest points, e.g. bisectors that have 2 or 3 closest points.

Ming Lin

Generalized Voronoi Diagrams



- The extension of the Voronoi diagram to higher dimensional features (such as edges and facets, instead of points); i.e. the set of points closest to a *feature*, e.g. that of a polyhedron.
- FACTS:
 - In general, the generalized Voronoi diagram has quadratic surface boundaries in it.
 - If the polyhedron is convex, then its generalized Voronoi diagram has planar boundaries.

Ming Lin

Voronoi Regions



- A *Voronoi region* associated with a *feature* is a set of points that are closer to that feature than any other.
- FACTS:
 - The Voronoi regions form a partition of space outside of the polyhedron according to the closest feature.
 - The collection of Voronoi regions of each polyhedron is the generalized Voronoi diagram of the polyhedron.
 - The generalized Voronoi diagram of a convex polyhedron has linear size and consists of polyhedral regions. And, all Voronoi regions are convex.

Ming Lin

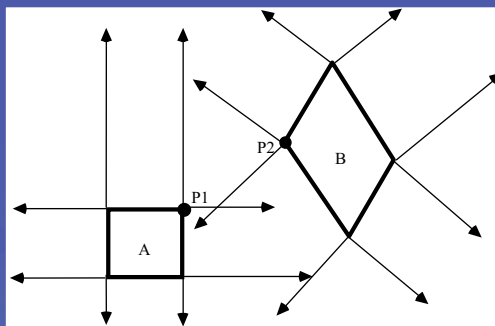
Voronoi Marching

Basic Ideas:

- Coherence: **local geometry does not change much, when computations repetitively performed over successive small time intervals**
- Locality: to "*track*" the pair of closest features between 2 moving convex polygons(polyhedra) w/ Voronoi regions
- Performance: **expected constant running time, independent of the geometric complexity**

Ming Lin

Simple 2D Example

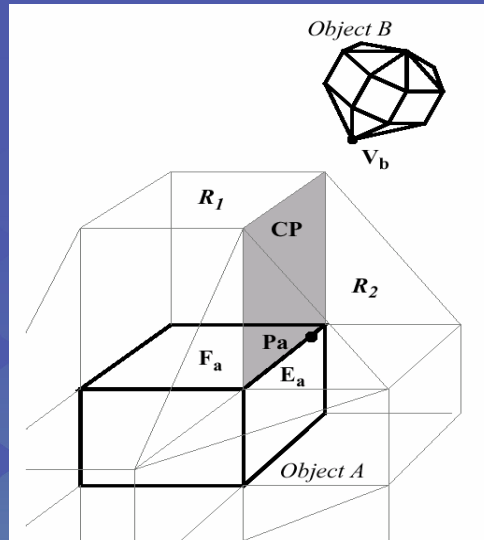


Objects A & B and their Voronoi regions: P1 and P2 are the pair of closest points between A and B. Note P1 and P2 lie within the Voronoi regions of each other.

Ming Lin

Basic Idea for Voronoi Marching


SIGGRAPH2004



Large, Dynamic Environments


SIGGRAPH2004

- For dynamic simulation where the velocity and acceleration of all objects are known at each step, use the scheduling scheme (implemented as heap) to prioritize "critical events" to be processed.
- Each object pair is tagged with the estimated time to next collision. Then, each pair of objects is processed accordingly. The heap is updated when a collision occurs.

Scheduling Scheme



- a_{max} : an upper bound on relative acceleration between any two *points* on any pair of objects.
- a_{lin} : relative absolute linear
- α : relative rotational accelerations
- ω : relative rotational velocities
- r : vector difference btw CoM of two bodies
- d : initial separation for two given objects

$$a_{max} = | a_{lin} + \alpha \times r + \omega \times \omega \times r |$$

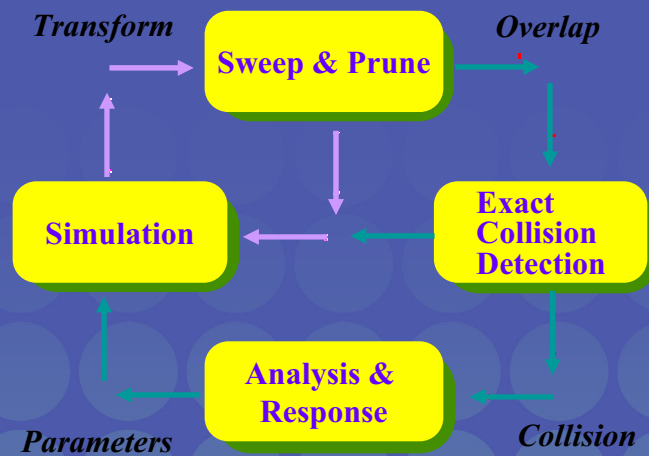
$$v_i = | v_{lin} + \omega \times r |$$

- Estimated Time to collision:

$$t_c = \{ (v_i^2 + 2 a_{max} d)^{1/2} - v_i \} / a_{max}$$

Ming Lin

Collide System Architecture



Ming Lin

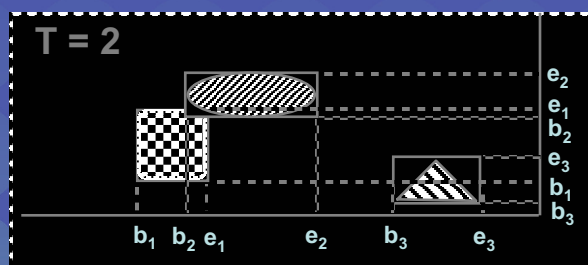
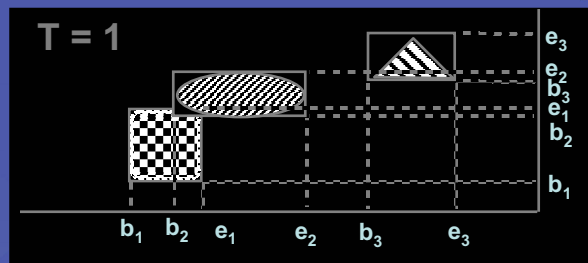
Sweep and Prune



- Compute the axis-aligned bounding box (fixed vs. dynamic) for each object
- Dimension Reduction by projecting boxes onto each x, y, z - axis
- Sort the endpoints and find overlapping intervals
- Possible collision -- only if projected intervals overlap in all 3 dimensions

Ming Lin

Sweep & Prune



Ming Lin

Updating Bounding Boxes



- Coherence (**greedy algorithm**)
- Convexity properties (**geometric properties of convex polytopes**)
- Nearly constant time, **if the motion is relatively "small"**

Ming Lin

Use of Sorting Methods



- **Initial sort** -- quick sort runs in $O(m \log m)$ just as in any ordinary situation
- **Updating** -- insertion sort runs in $O(m)$ due to coherence. We sort an almost sorted list from last stimulation step. In fact, we look for "swap" of positions in all 3 dimension.

Ming Lin

Implementation Issues



- Collision matrix -- basically *adjacency matrix*
- Enlarge bounding volumes with some tolerance threshold
- Quick start polyhedral collision test -- using bucket sort & look-up table

Ming Lin

References



- [Collision Detection between Geometric Models: A Survey](#), by M. Lin and S. Gottschalk, Proc. of IMA Conference on Mathematics of Surfaces 1998.
- *I-COLLIDE: Interactive and Exact Collision Detection for Large-Scale Environments*, by Cohen, Lin, Manocha & Ponamgi, Proc. of ACM Symposium on Interactive 3D Graphics, 1995.
(More details in Chapter 3 of M. Lin's Thesis)

Ming Lin