Reading Assignments

- *Interactive Collision Detection, by P. M. Hubbard, Proc. of IEEE Symposium on Research Frontiers in Virtual Reality, 1993.*
- *Evaluation of Collision Detection Methods for Virtual Reality Fly-Throughs*, by Held, Klosowski and Mitchell, Proc. of Canadian Conf. on Computational Geometry 1995.
- *Efficient collision detection using bounding volume hierarchies of k-dops*, by J. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan, IEEE Trans. on Visualization and Computer Graphics, 4(1):21--37, 1998.

Reading Assignments

- ***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)**

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

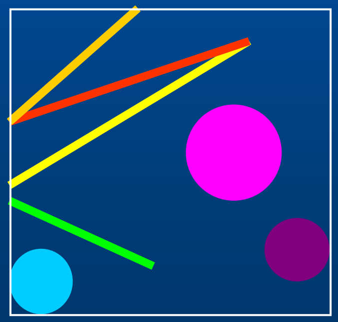
Hierarchical Representations

- 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

BVH vs. Spatial Partitioning

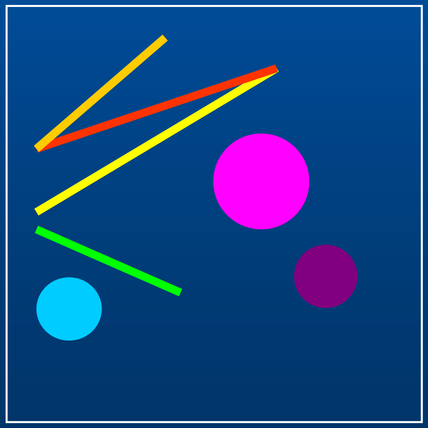
BVH:

- Object centric
- Spatial redundancy



SP:

- Space centric
- Object redundancy



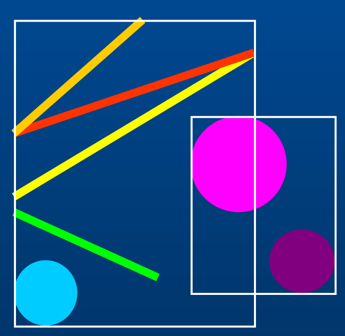
BVH vs. Spatial Partitioning

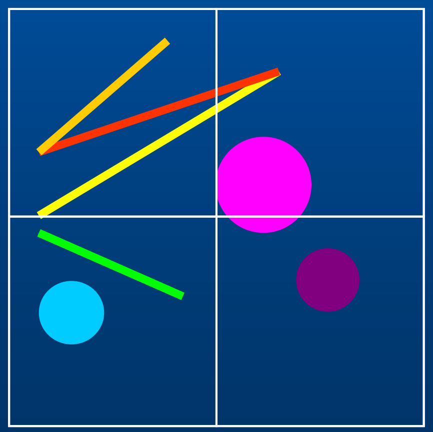
BVH:

- Object centric
- Spatial redundancy

SP:

- Space centric
- Object redundancy





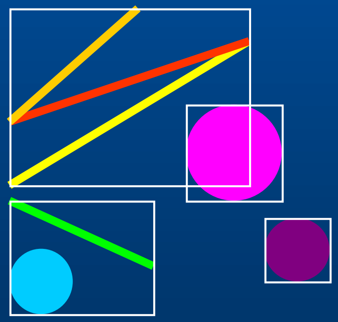
BVH vs. Spatial Partitioning

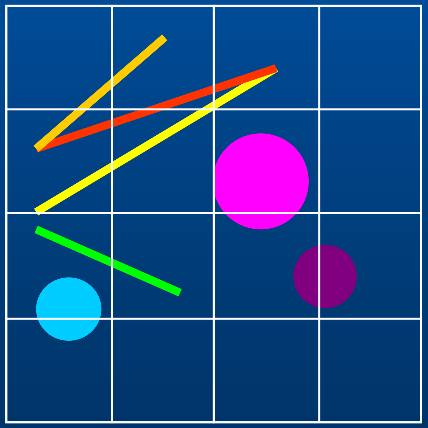
BVH:

- Object centric
- Spatial redundancy

SP:

- Space centric
- Object redundancy





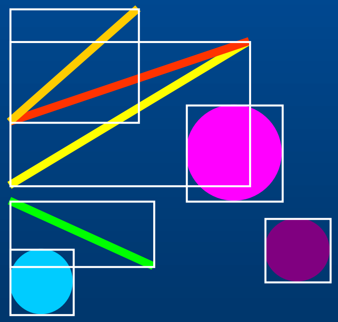
BVH vs. Spatial Partitioning

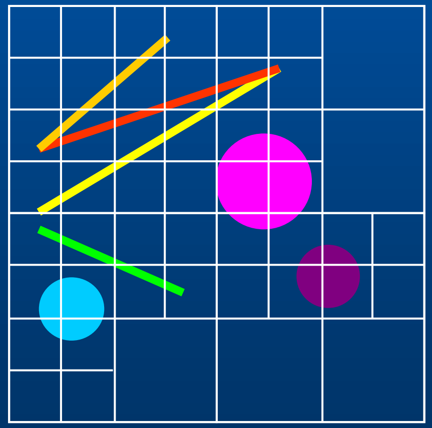
BVH:

- Object centric
- Spatial redundancy

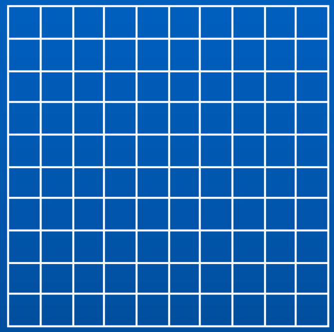
SP:

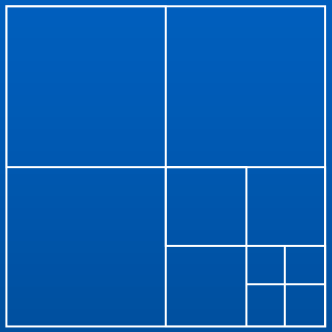
- Space centric
- Object redundancy

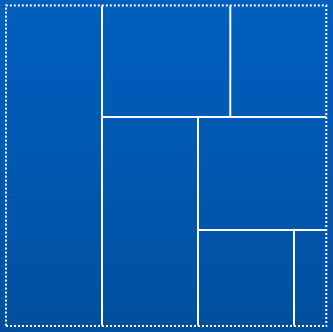


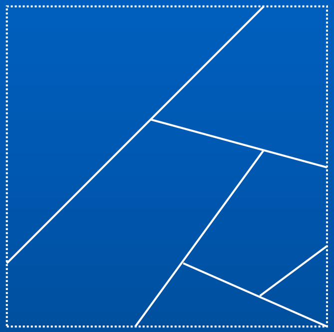


Spatial Data Structures & Subdivision


Uniform Spatial Sub


Quadtree/Octree


kd-tree

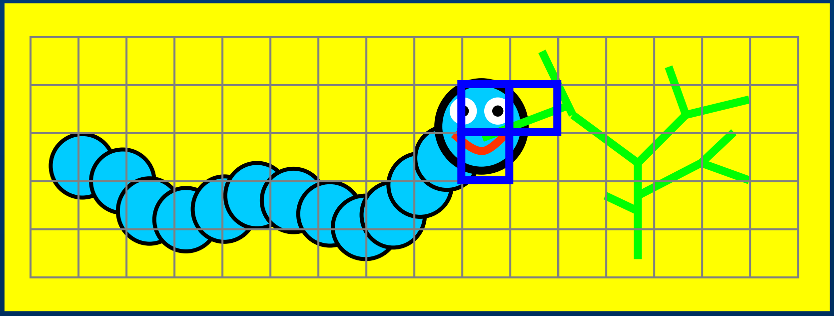

BSP-tree

- **Many others……**

  (see the lecture notes)

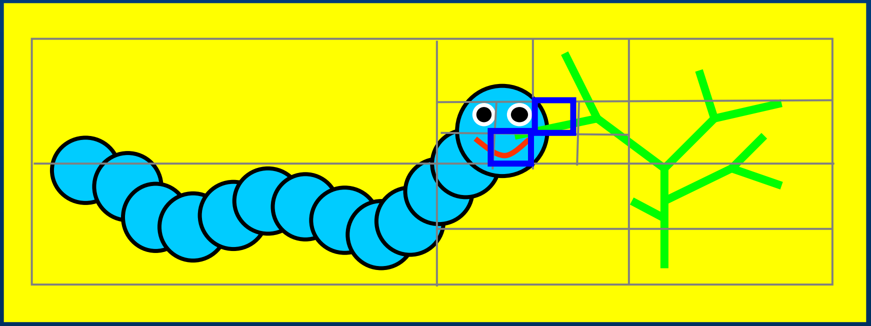
Uniform Spatial Subdivision

- 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"



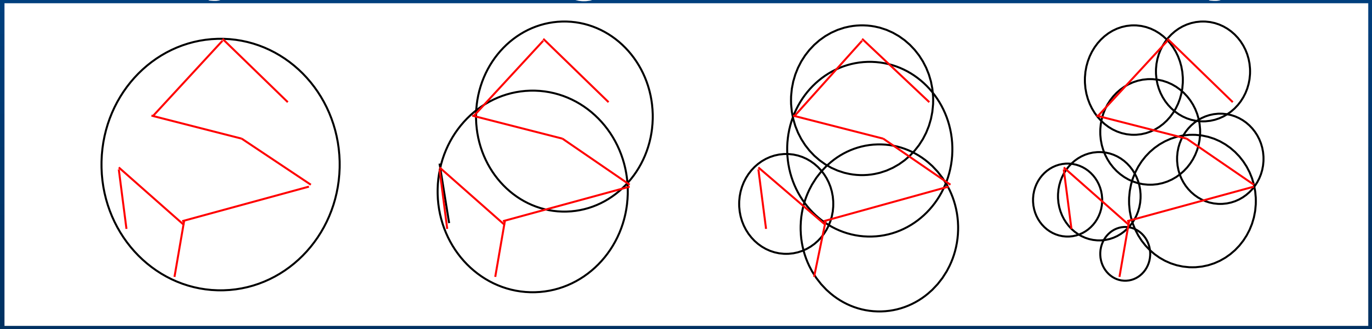
Octrees

- *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

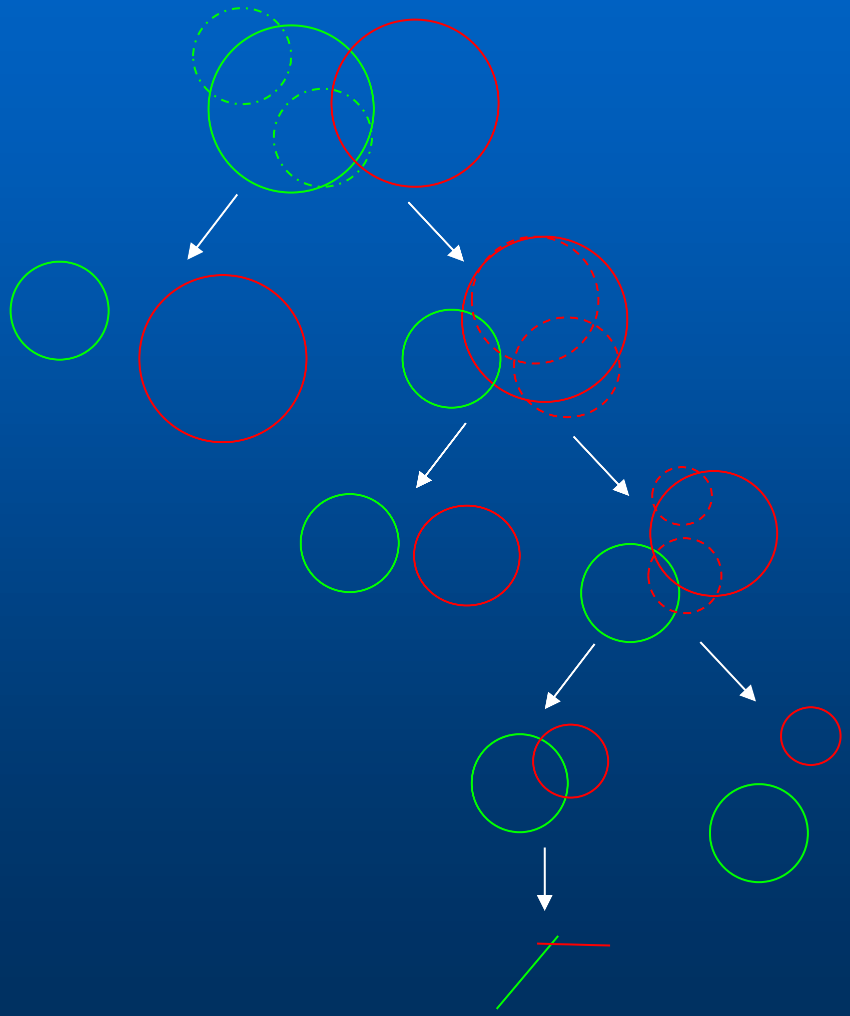
- **Model Hierarchy:**
  - each node has a simple volume that bounds a set of triangles
  - children contain volumes that each bound a different portion of the parent's triangles
  - The leaves of the hierarchy usually contain individual triangles
- **A binary bounding volume hierarchy:**



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 Swetp Spheres (PSS)
  - Line Swept Spheres (LSS)
  - Rectangle Swept Spheres (RSS)
  - Triangle Swept Spheres (TSS)

BVH-Based Collision Detection



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"

Evaluating Bounding Volume Hierarchies

Cost Function:

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

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

Designing Bounding Volume Hierarchies

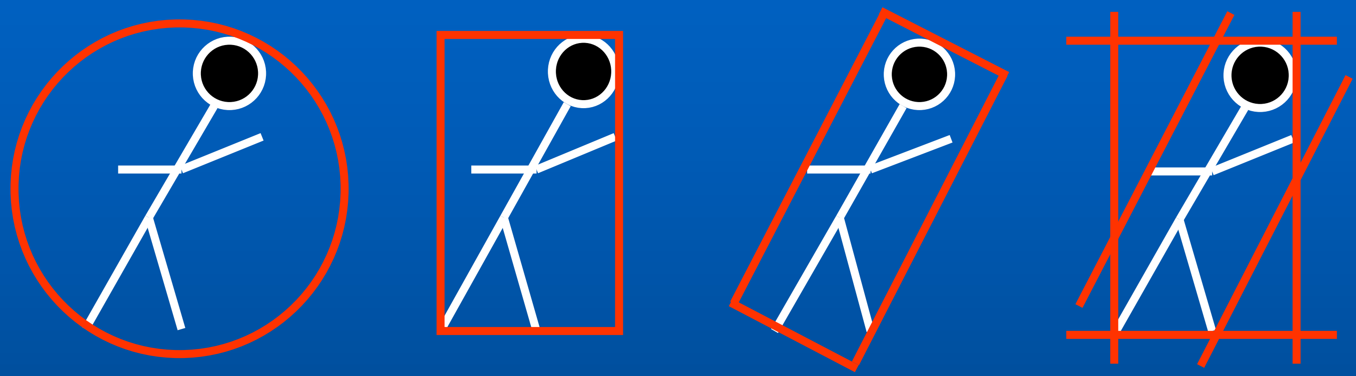
The choice governed by these constraints:

- 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)

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.

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)

Building Hierarchies

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

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

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……

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.

Choices of k-dops in 3D

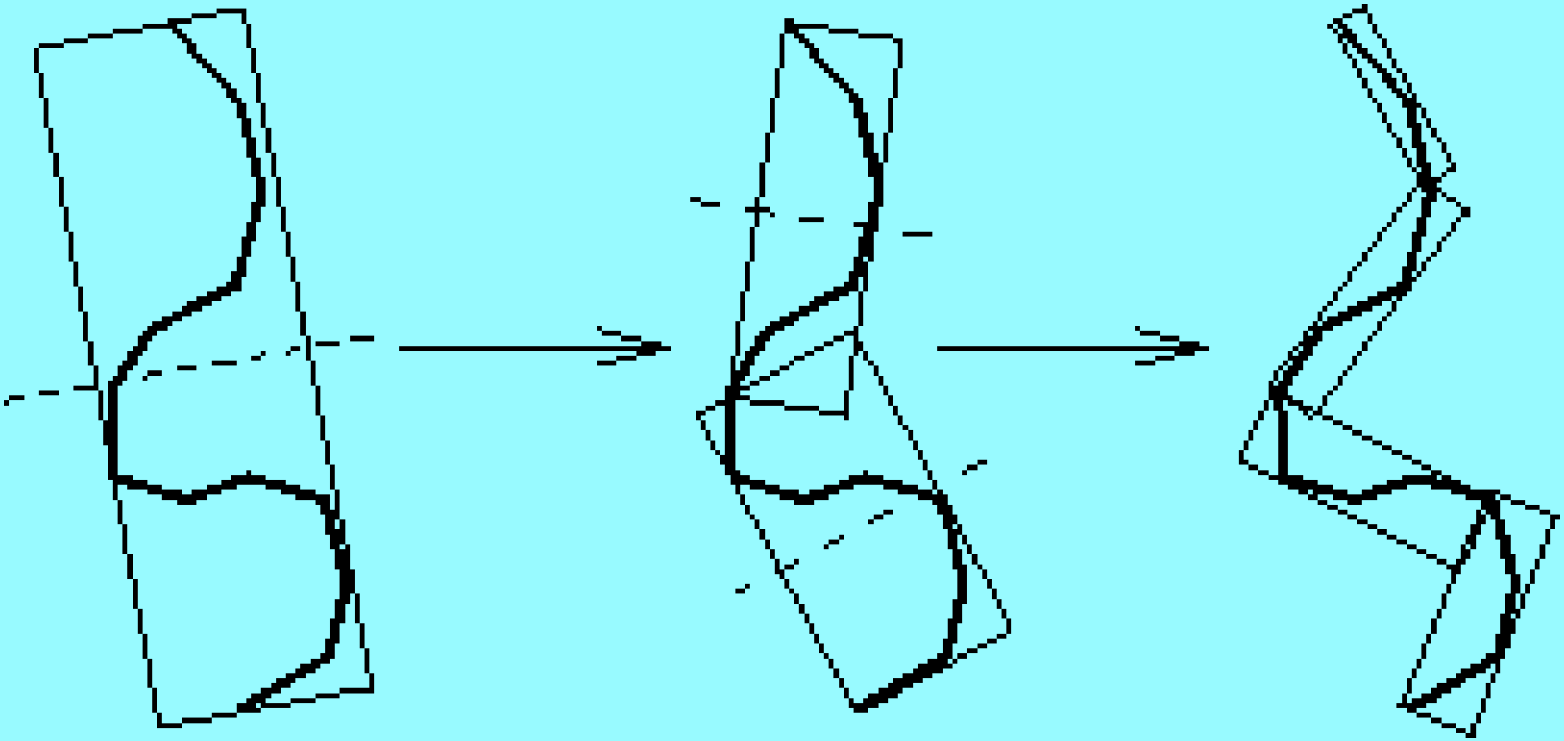
- 6-dop: defined by coordinate axes
- 14-dop: defined by the vectors (1,0,0), (0,1,0), (0,0,1), (1,1,1), (1,-1,1), (1,1,-1) and (1,-1,-1)
- 18-dop: defined by the vectors (1,0,0), (0,1,0), (0,0,1), (1,1,0), (1,0,1), (0,1,1), (1,-1,0), (1,0,-1) and (0,1,-1)
- 26-dop: defined by the vectors (1,0,0), (0,1,0),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

Building Trees of k-dops

The major issue is updating the *k*-dops:

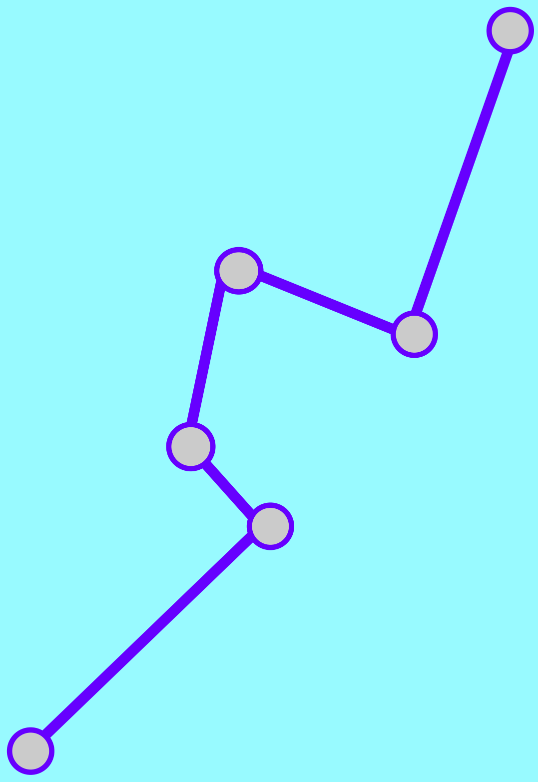
- **Use Hill Climbing (as proposed in I-Collide) to update the min/max along each *k/2* directions by comparing with the neighboring vertices**
- **But, the object may not be convex…… Use the approximation (convex hull vs. another *k*-dop)**

Building an OBBTree



Recursive top-down construction:
partition and refit

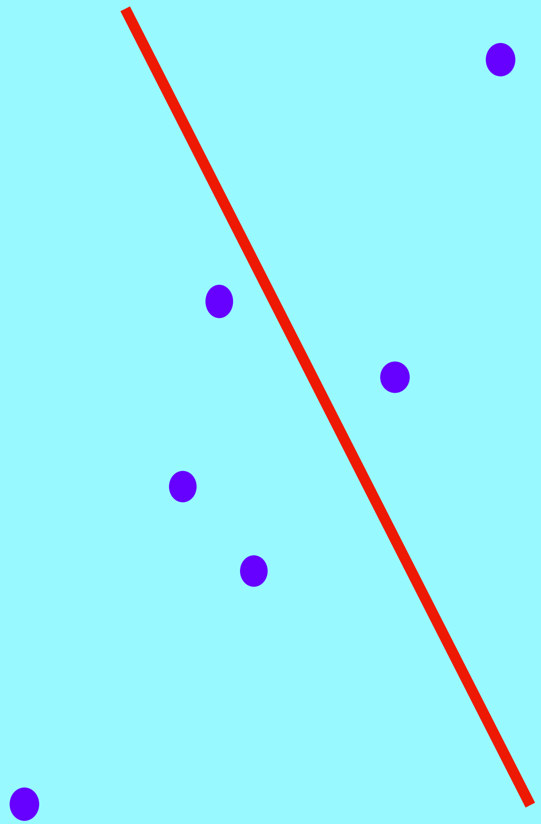
Building an OBB Tree



Given some polygons, consider their vertices...

Building an OBB Tree

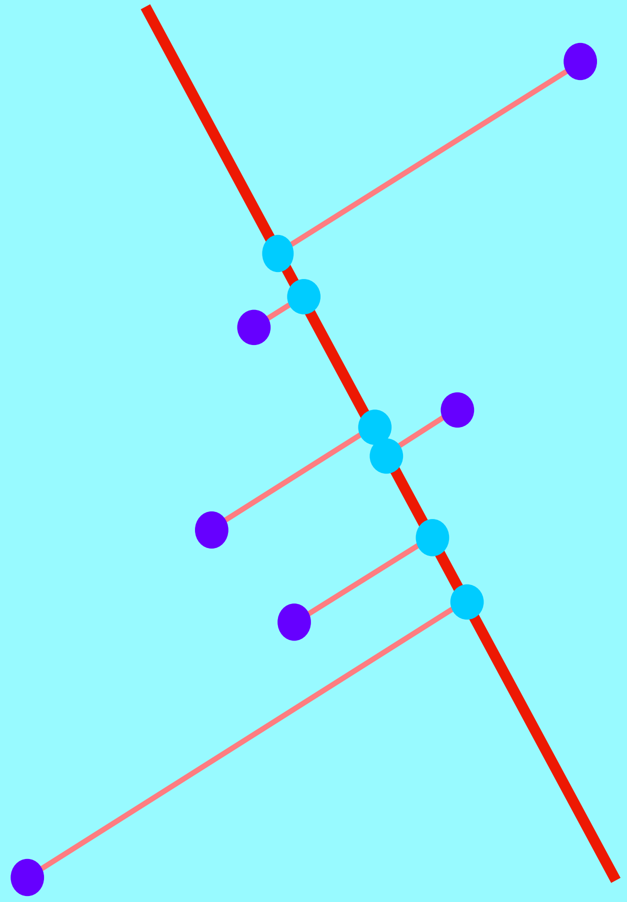
… and an arbitrary line



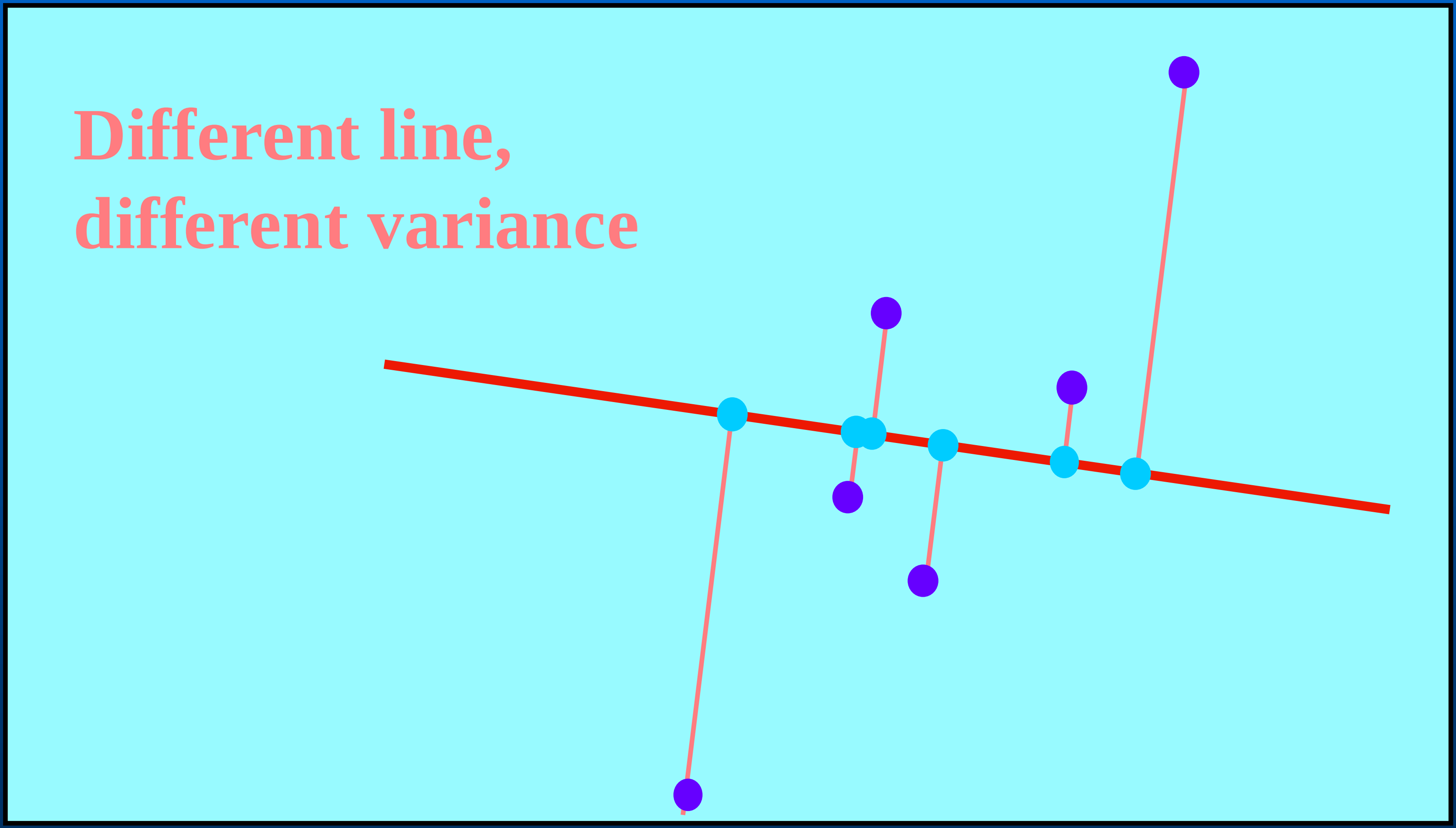
Building an OBB Tree

Project onto the line

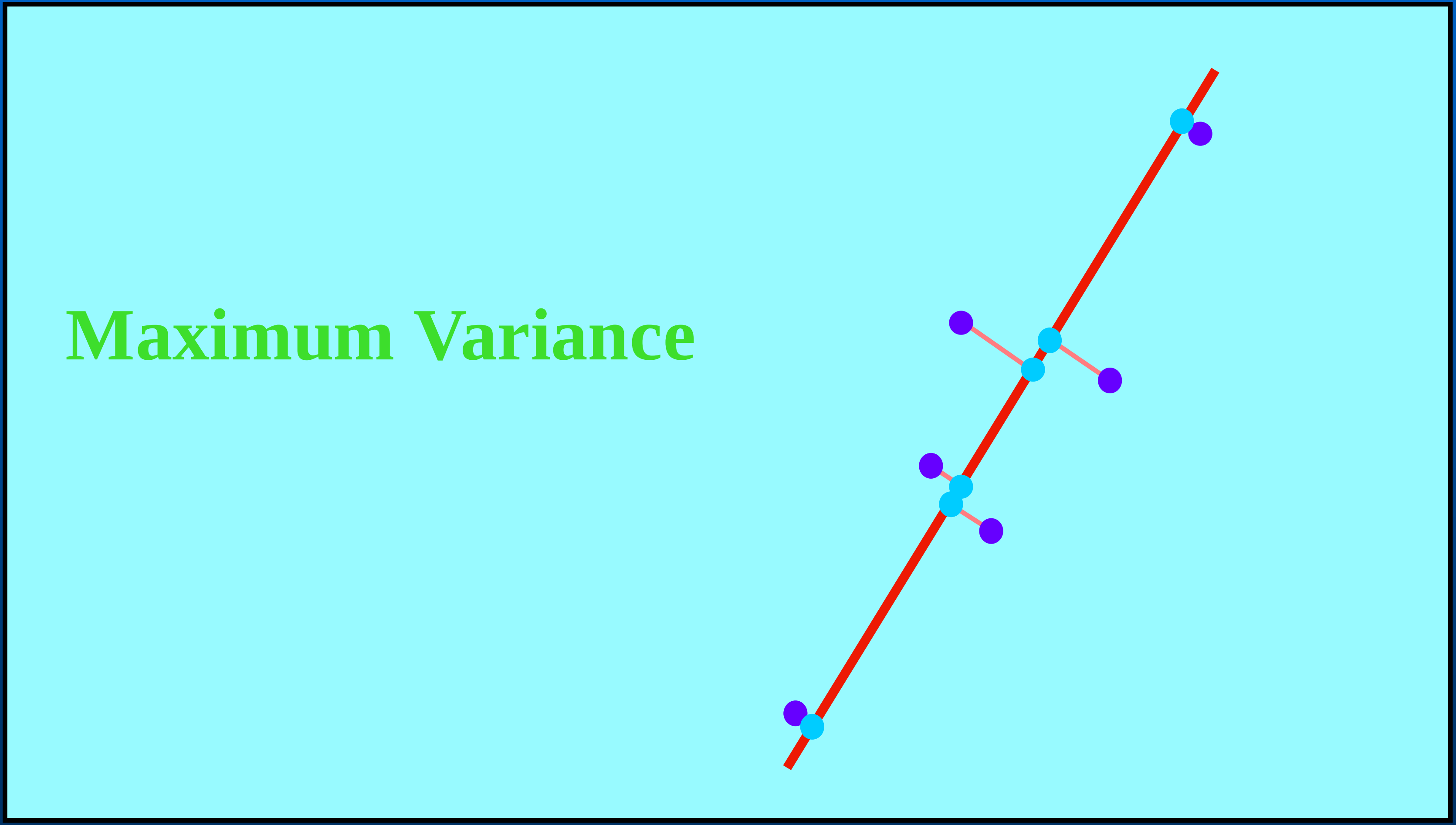
Consider variance of distribution on the line



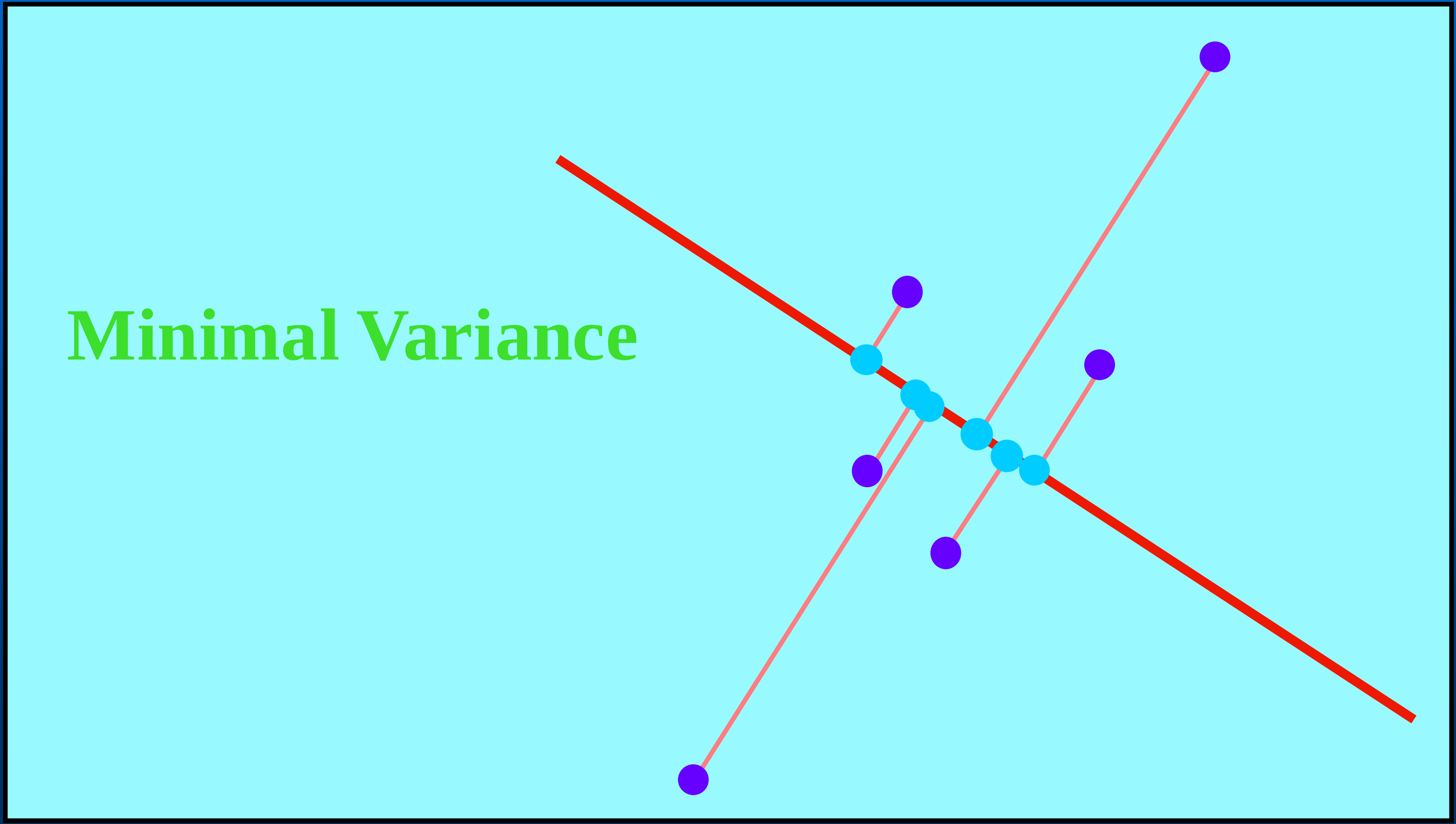
Building an OBB Tree



Building an OBB Tree

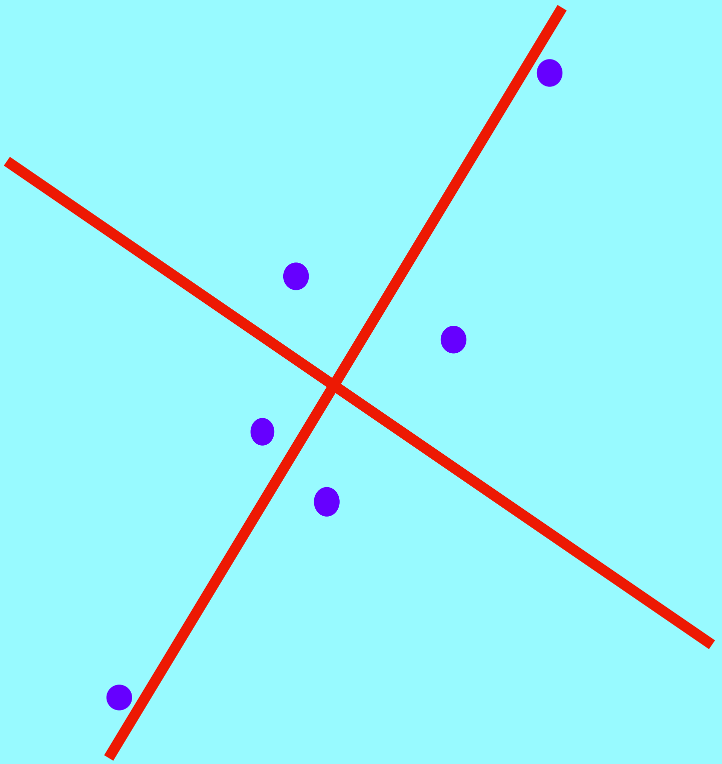


Building an OBB Tree



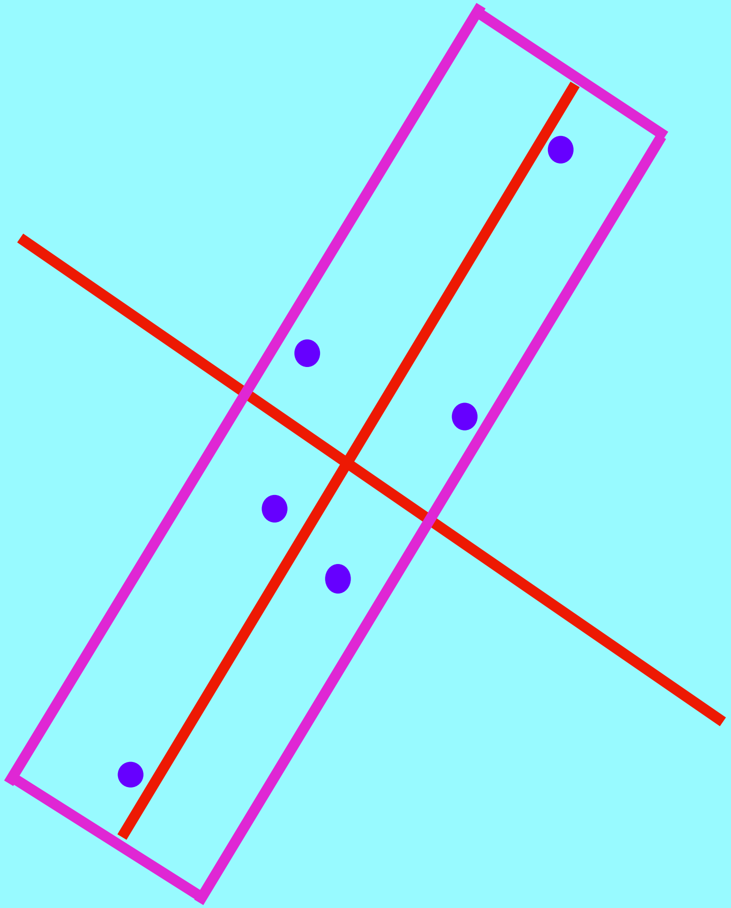
Building an OBB Tree

Given by eigenvectors of covariance matrix of coordinates of original points

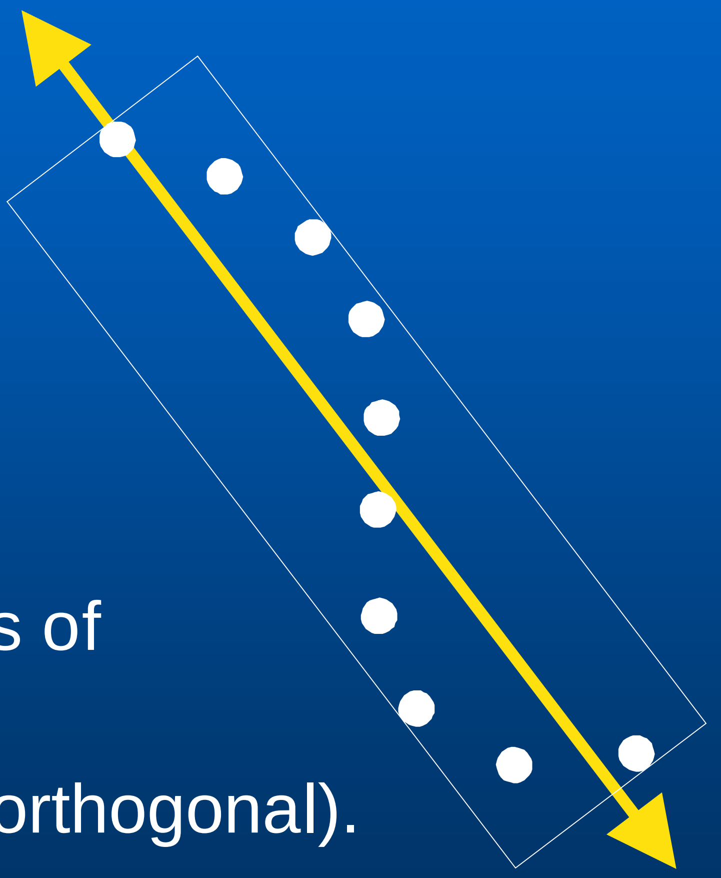


Building an OBB Tree

Choose bounding box oriented this way



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).

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 $\boldsymbol{C}$ can be expressed in vector notation as:

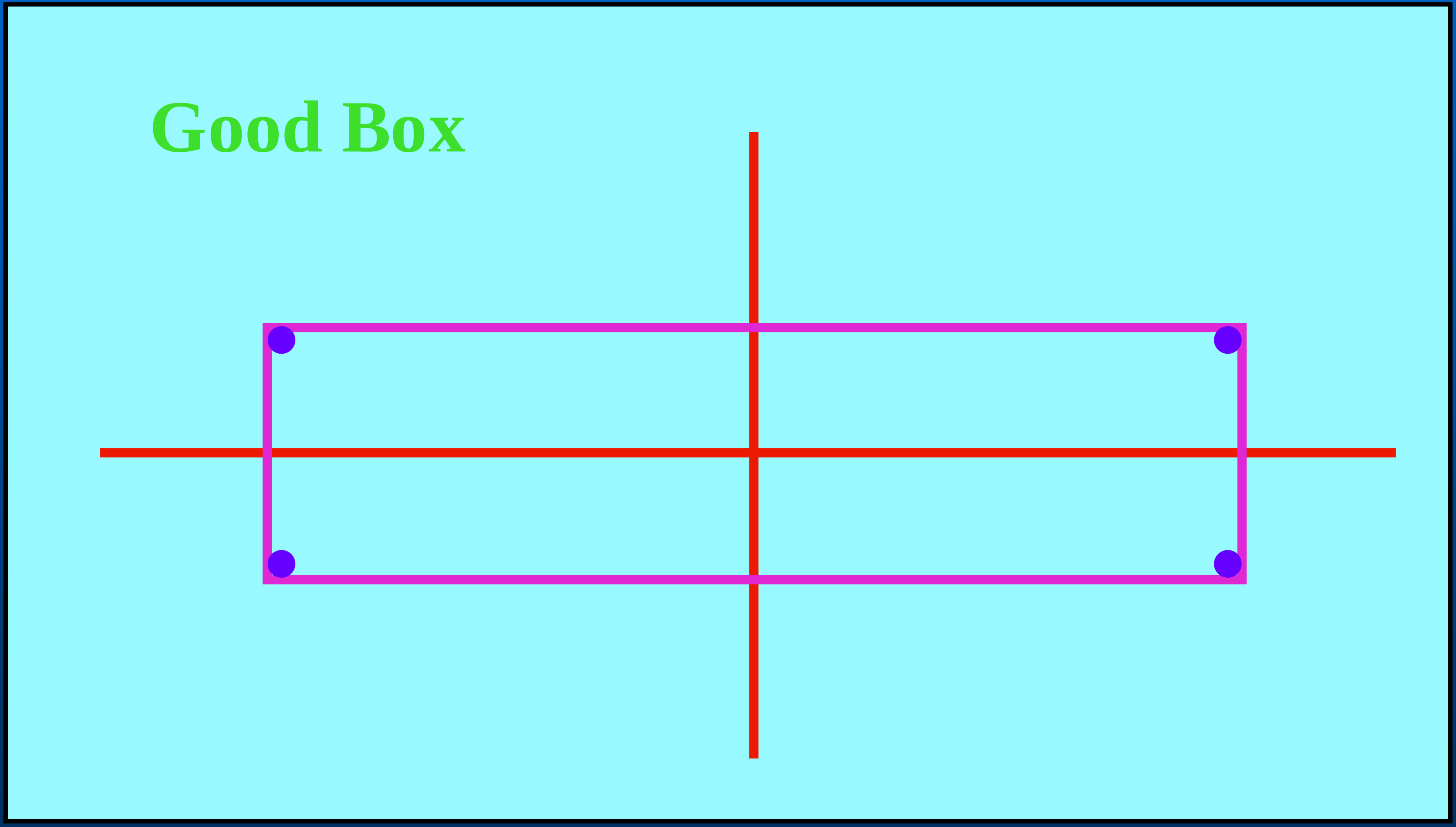
$$\mu = \frac{1}{3n} \sum_{i=0}^{n} (\mathbf{a}^i + \mathbf{b}^i + \mathbf{c}^i),$$

$$\mathbf{C}_{jk} = \frac{1}{3n} \sum_{i=0}^{n} \left(\overline{\mathbf{a}}^i_j \overline{\mathbf{a}}^i_k + \overline{\mathbf{b}}^i_j \overline{\mathbf{b}}^i_k + \overline{\mathbf{c}}^i_j \overline{\mathbf{c}}^i_k \right), \qquad 1 \le j, k \le 3$$

where n is the number of triangles, and

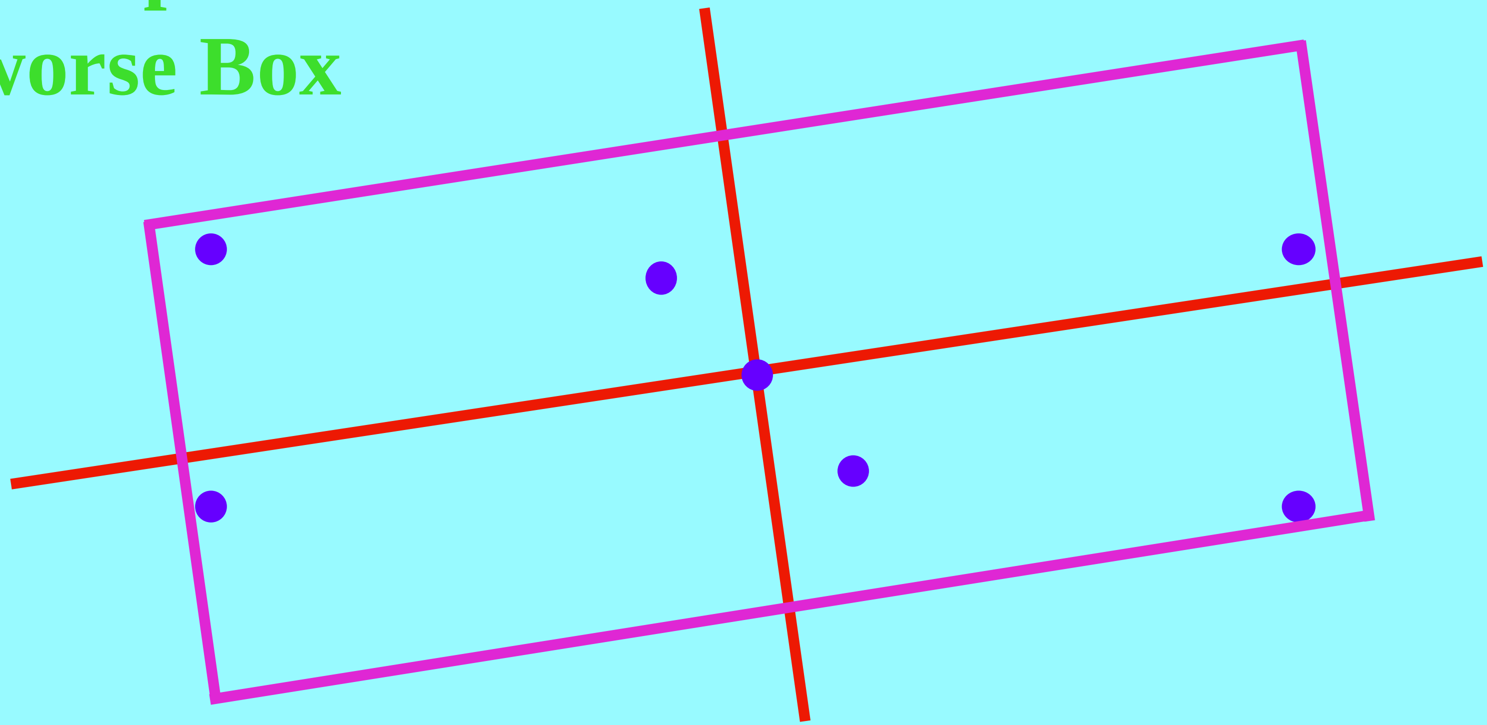
$$\overline{\mathbf{a}}^i = \mathbf{a}^i - \mu, \ \overline{\mathbf{b}}^i = \mathbf{b}^i - \mu, \ \overline{\mathbf{c}}^i = \mathbf{c}^i - \mu.$$

Building an OBB Tree

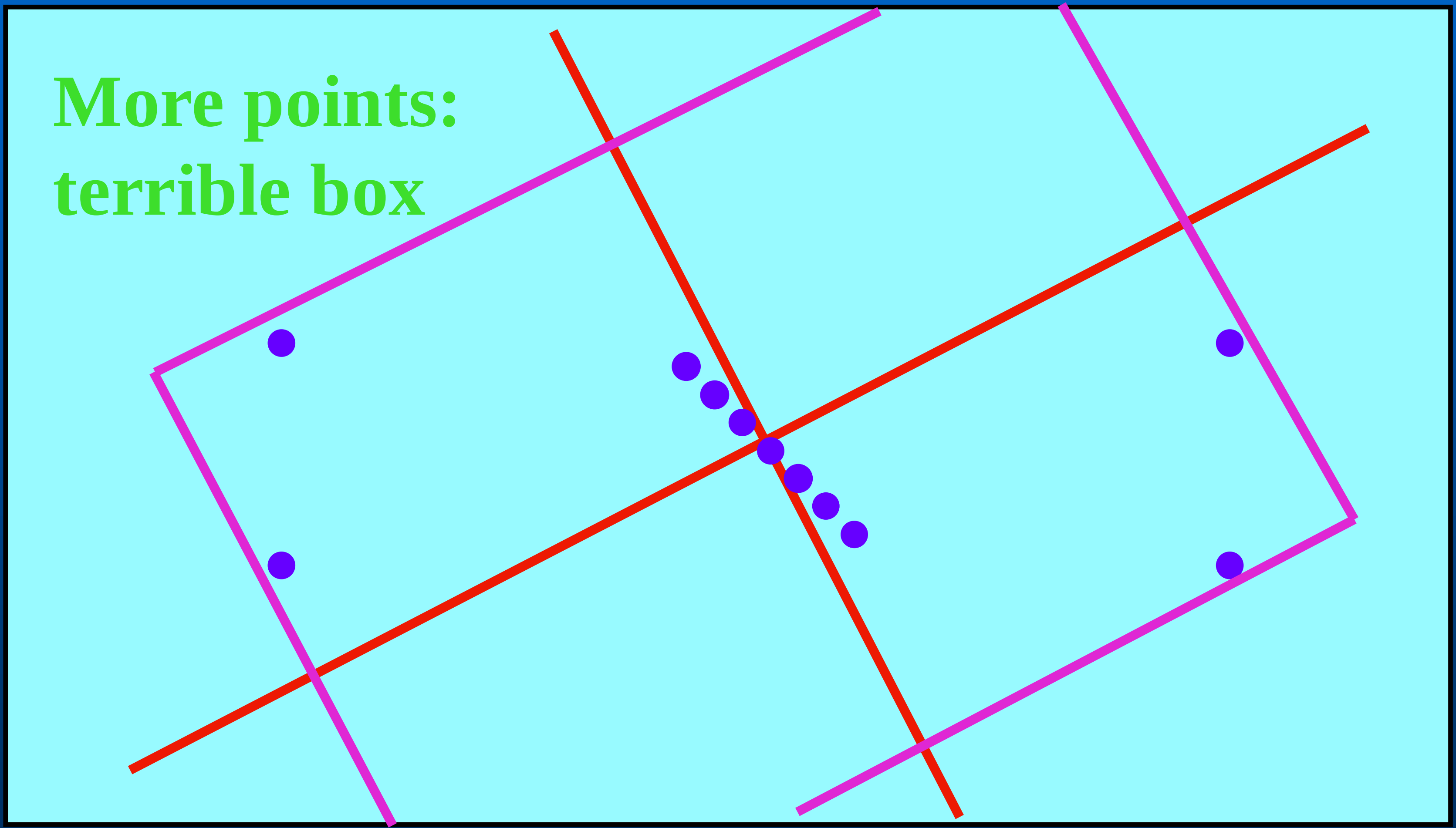


Building an OBB Tree

Add points:
worse Box

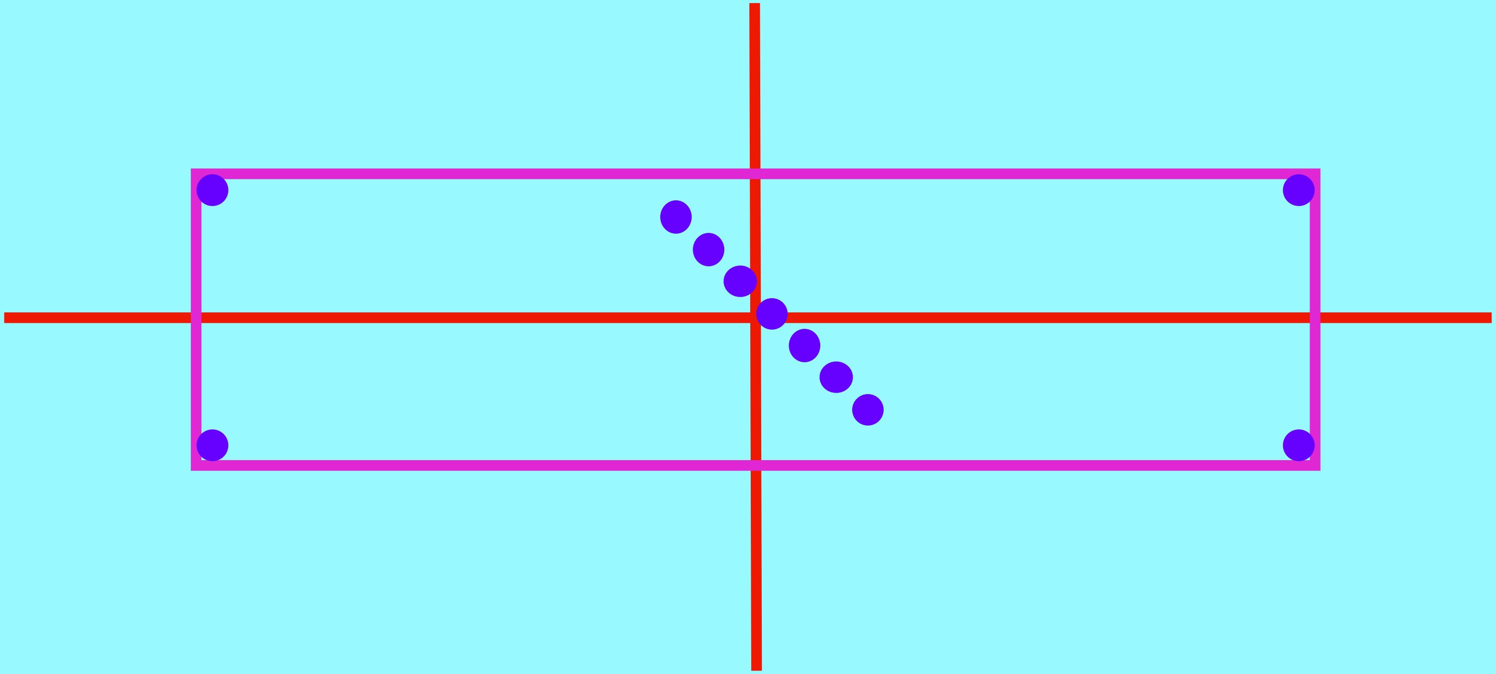


Building an OBB Tree



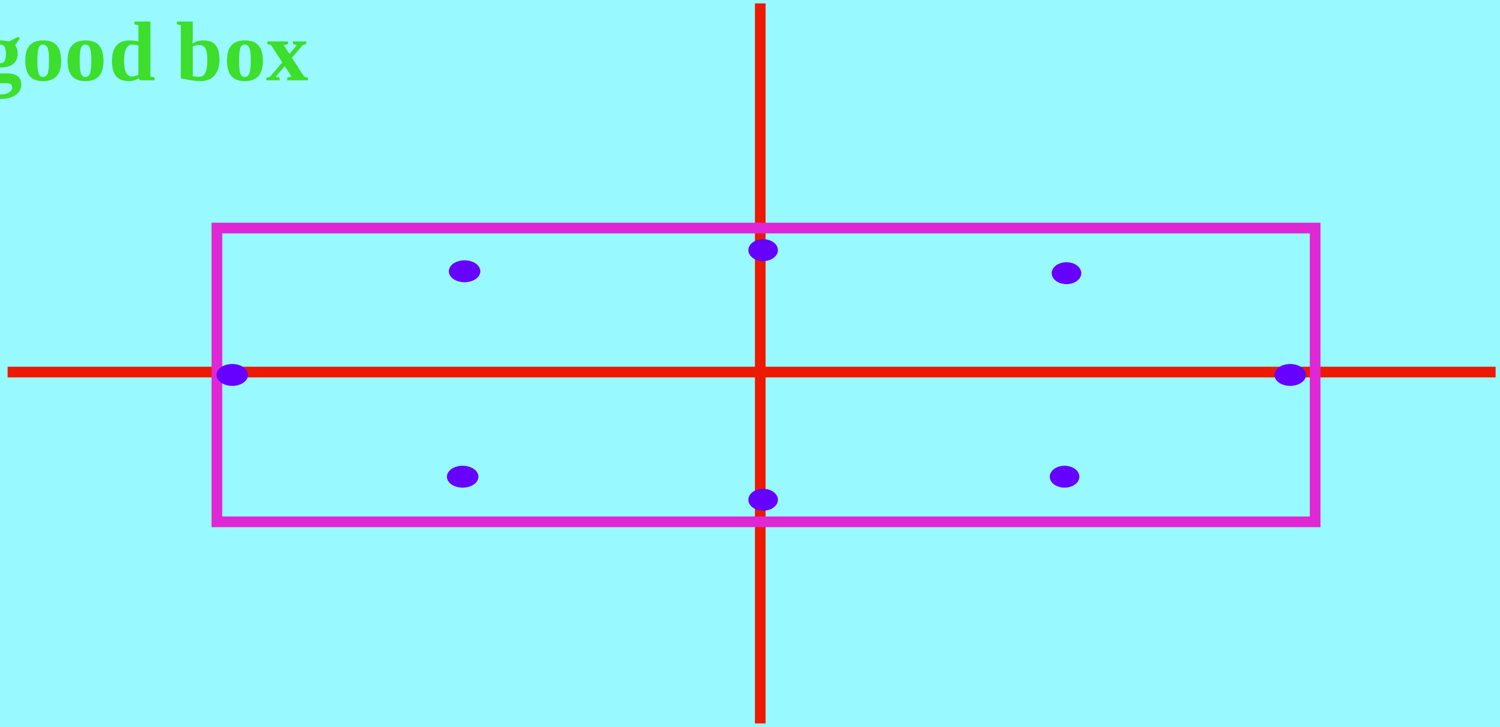
Building an OBB Tree

Compute with extremal points only



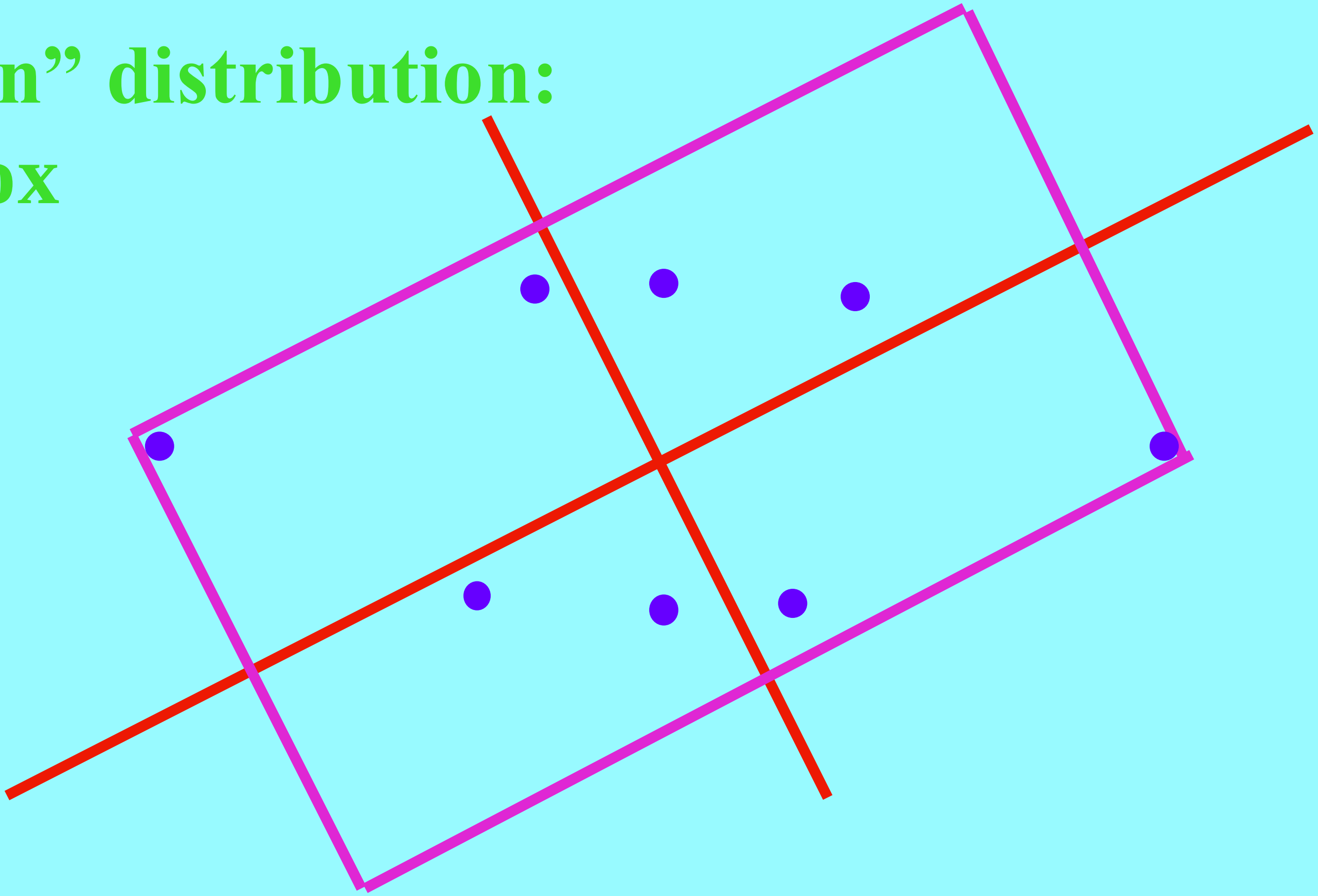
Building an OBB Tree

“Even” distribution:
good box



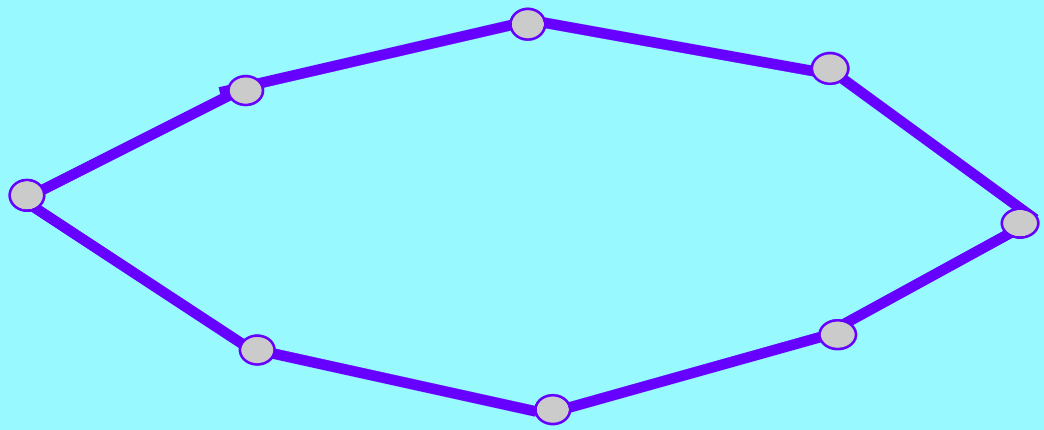
Building an OBB Tree

"Uneven" distribution:
bad box



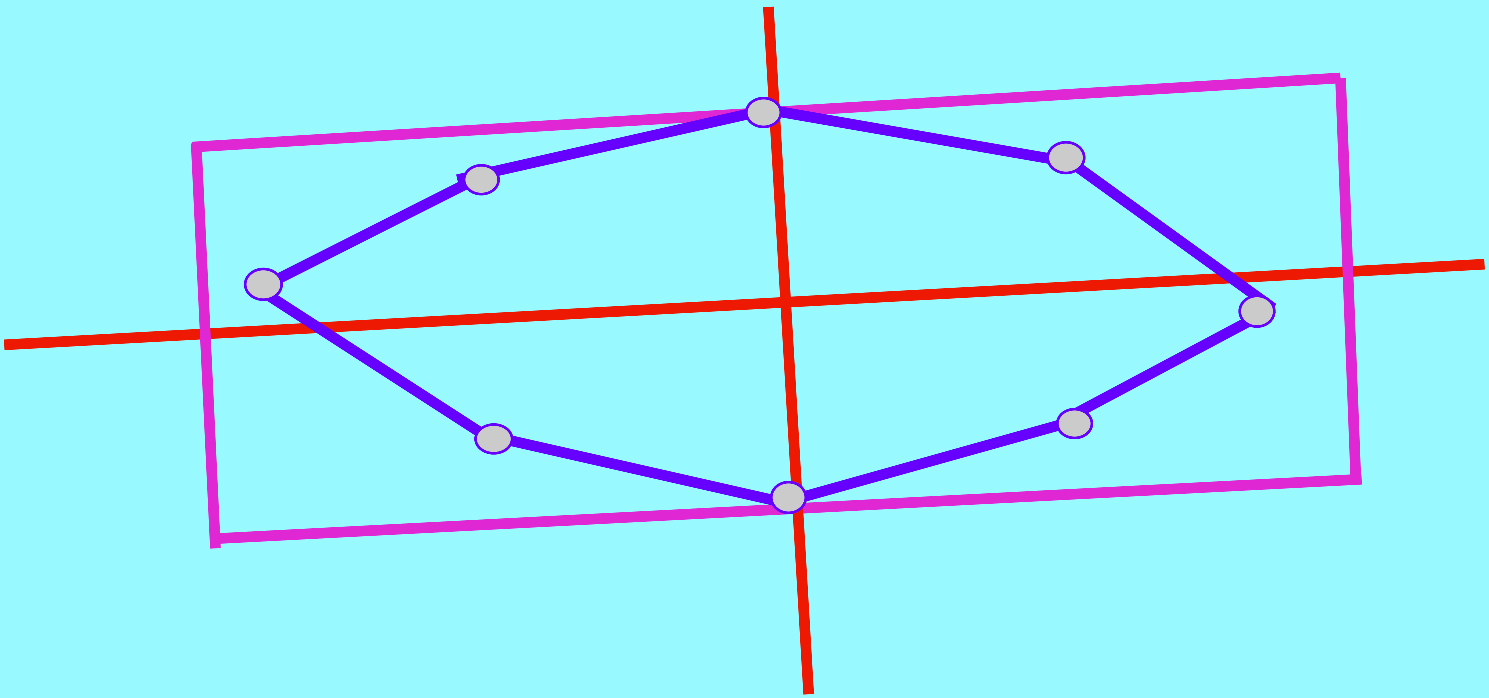
Building an OBB Tree

Fix: Compute facets of convex hull...

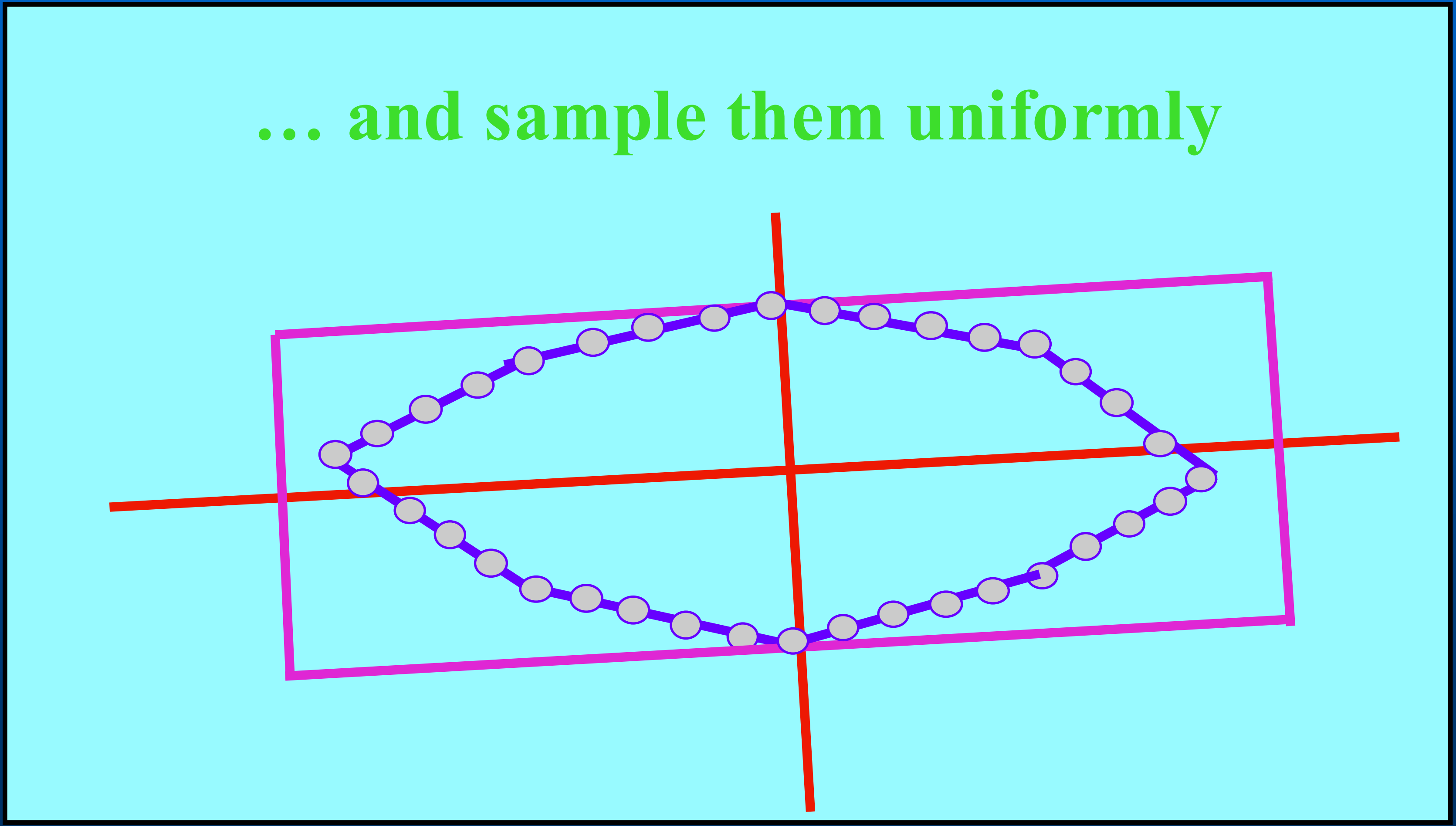


Building an OBB Tree

Better: Integrate over facets



Building an OBB Tree



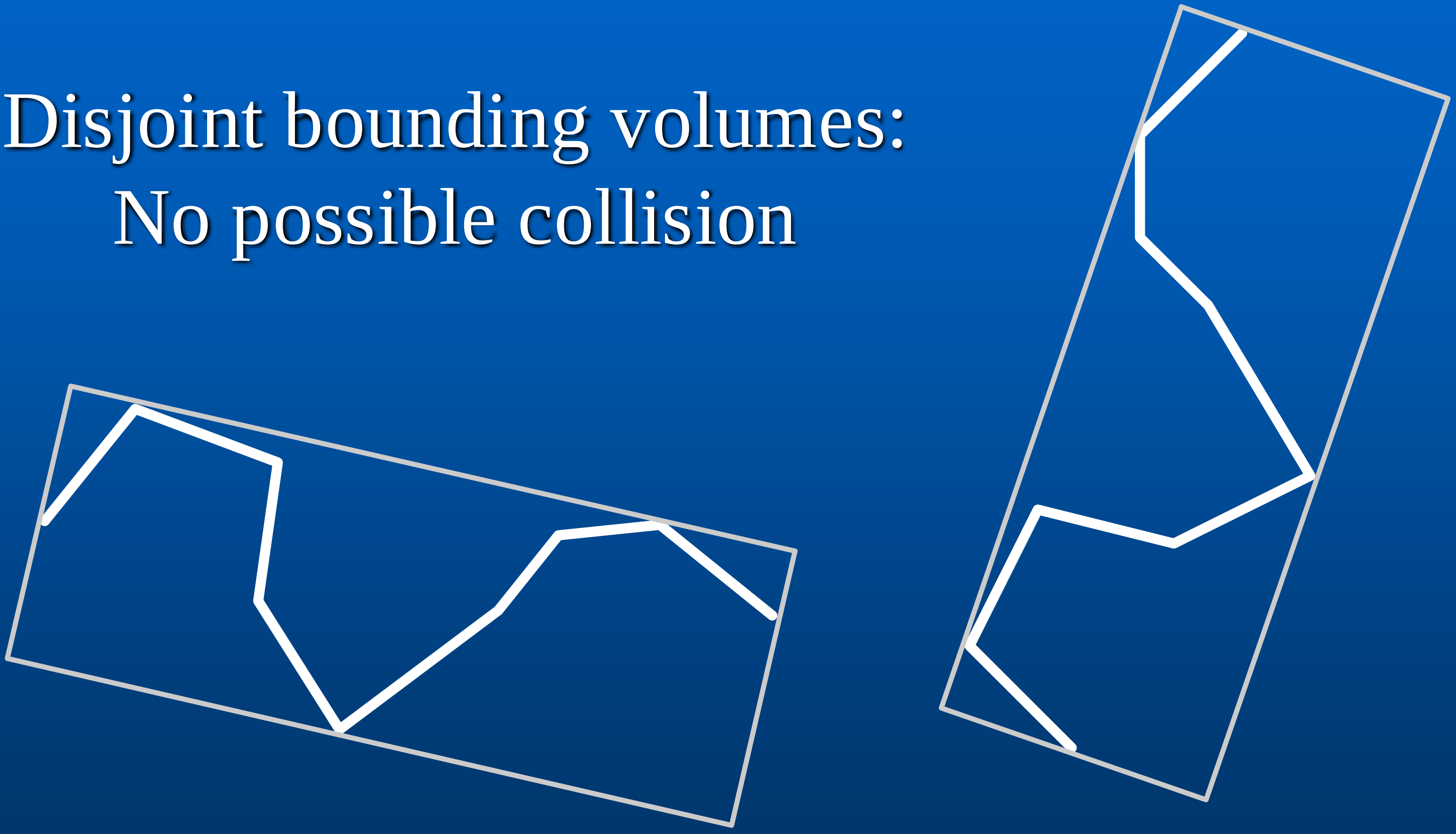
Building an OBB Tree: Summary

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

Tree Traversal

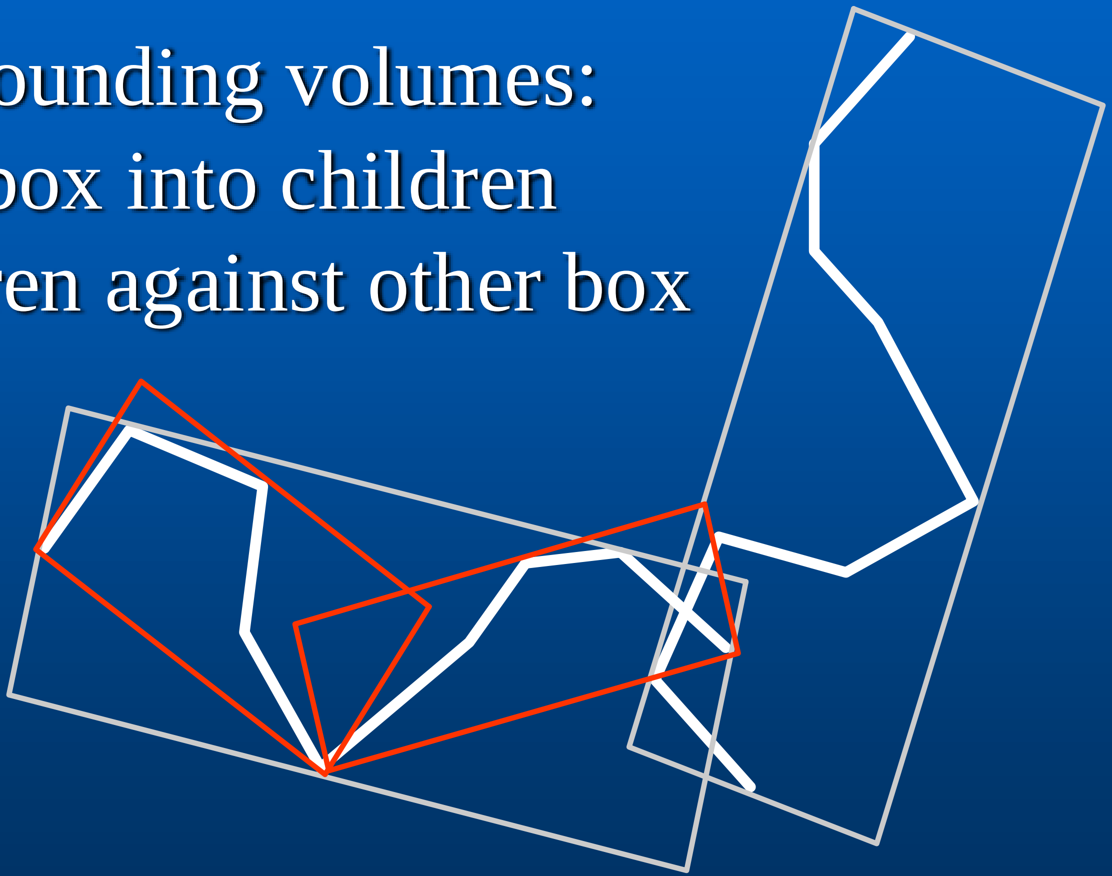
Disjoint bounding volumes:
No possible collision



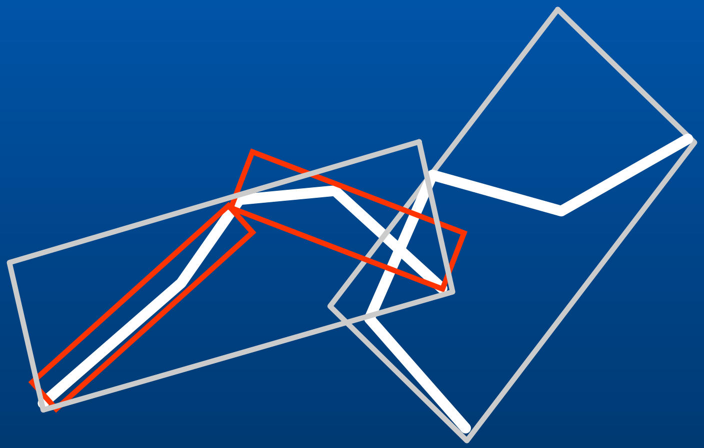
Tree Traversal

Overlapping bounding volumes:

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

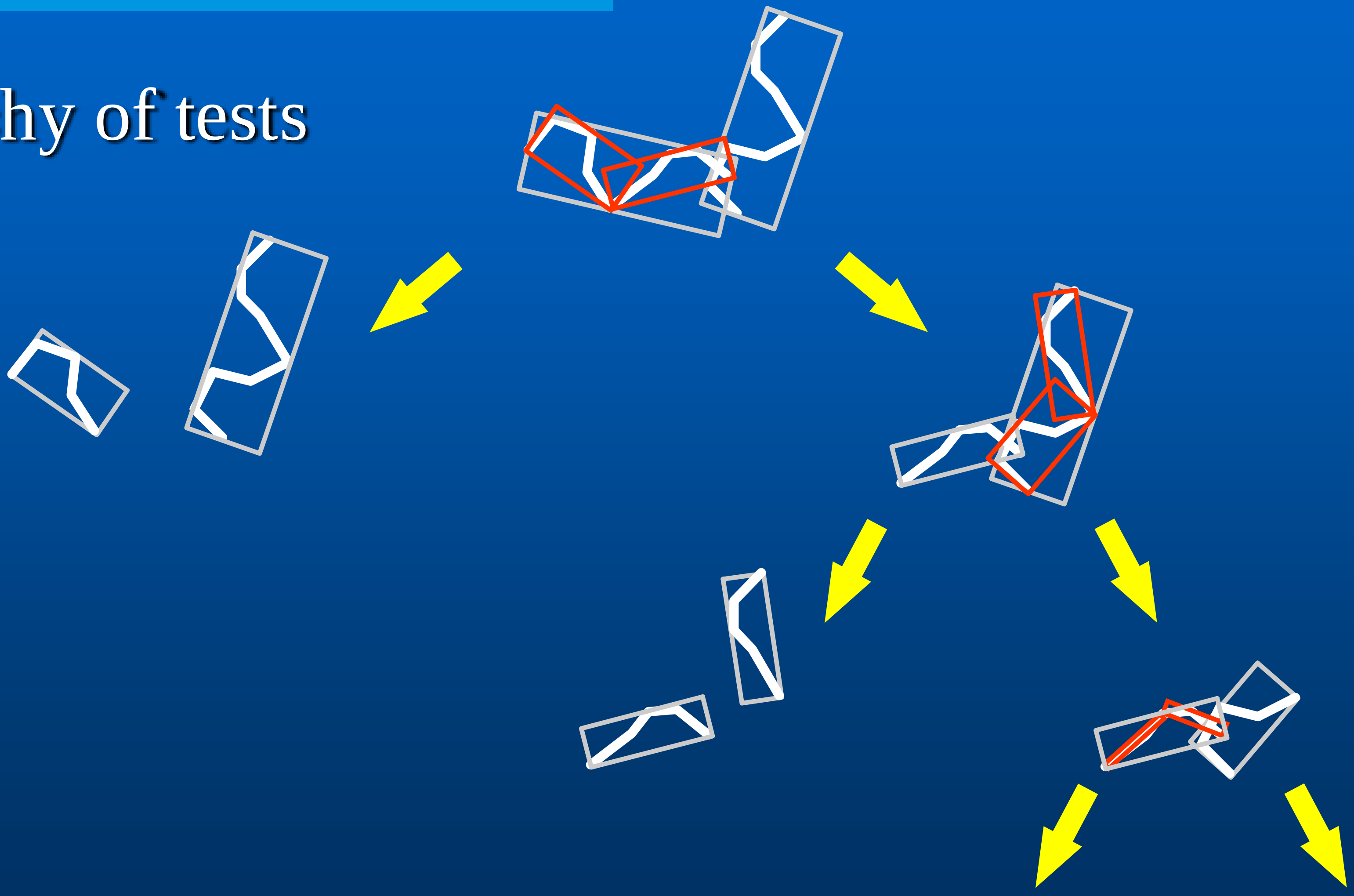


Tree Traversal

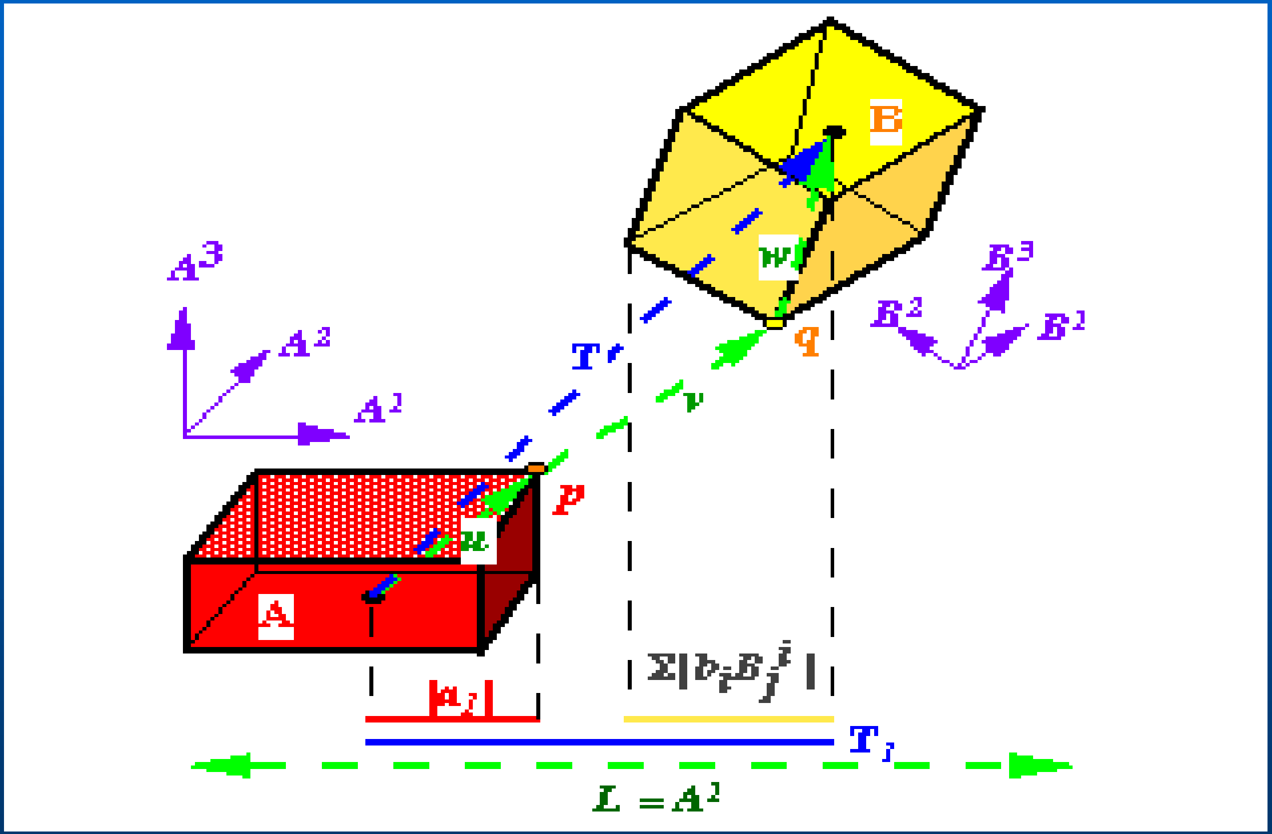


Tree Traversal

- Hierarchy of tests



Separating Axis Theorem



- L is a separating axis for OBBs A & B, since A & B become disjoint intervals under projection onto L

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

perpedicular to an edge from each

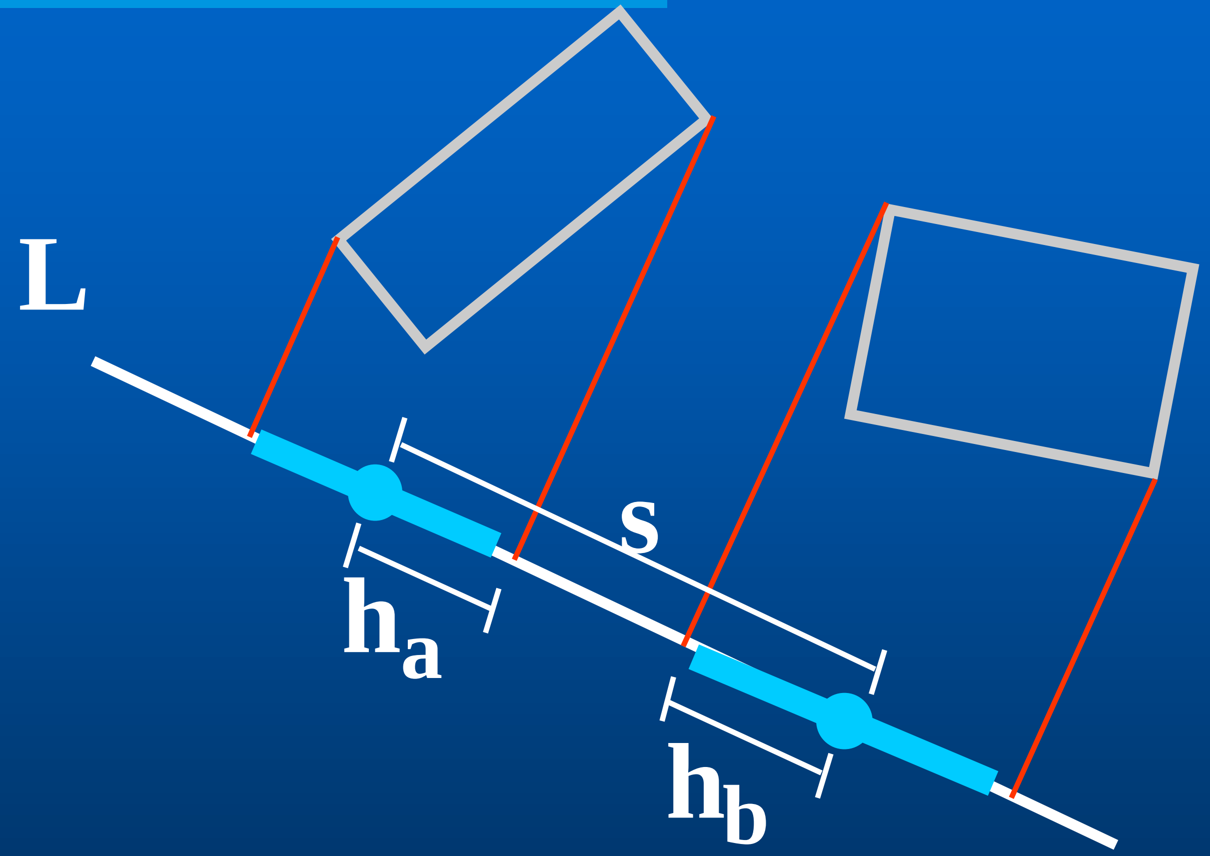
Implications of Theorem

- 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.

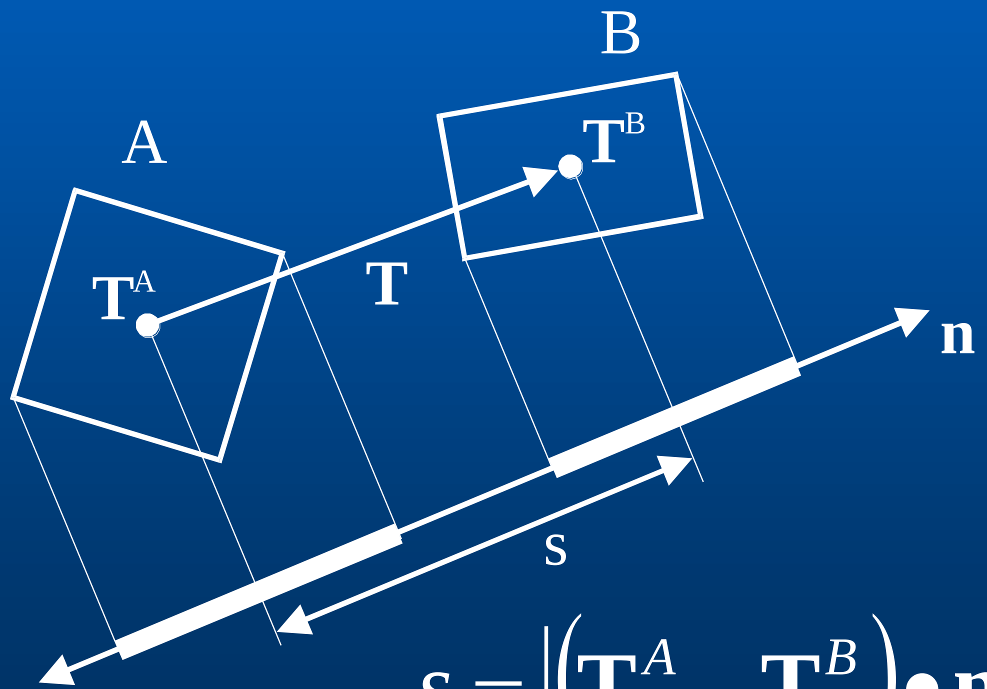
OBB Overlap Test: An Axis Test



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

OBB Overlap Test: Axis Test Details

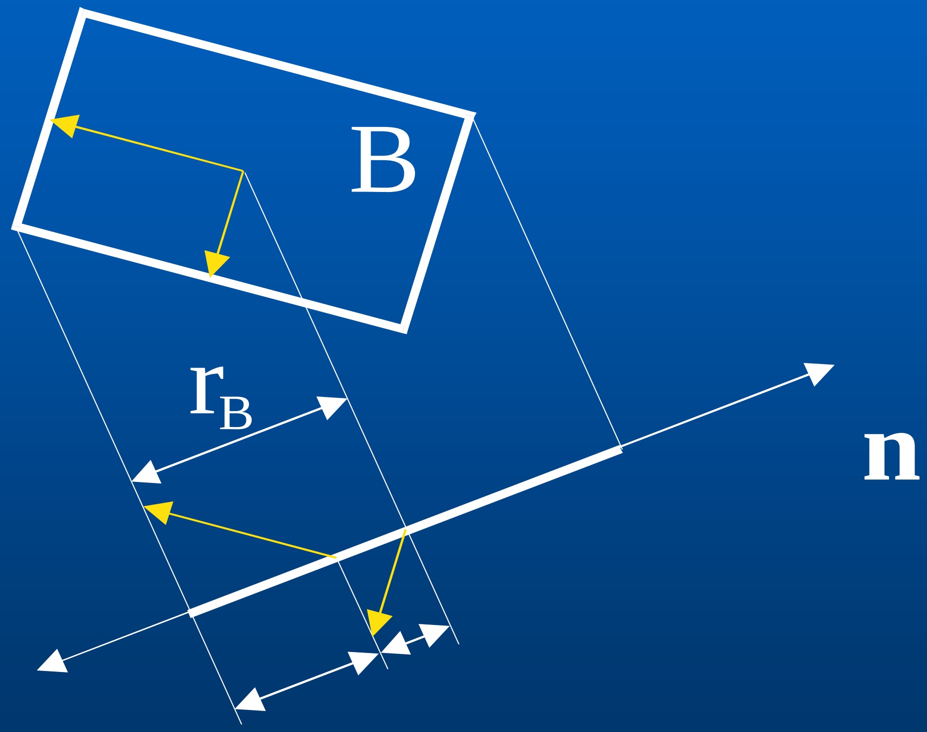
Box centers project to interval midpoints, so midpoint separation is length of vector T's image.



$$s = \left| \left(\mathbf{T}^A - \mathbf{T}^B \right) \bullet \mathbf{n} \right|$$

OBB Overlap Test: Axis Test Details

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



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

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.

OBB Overlap Test

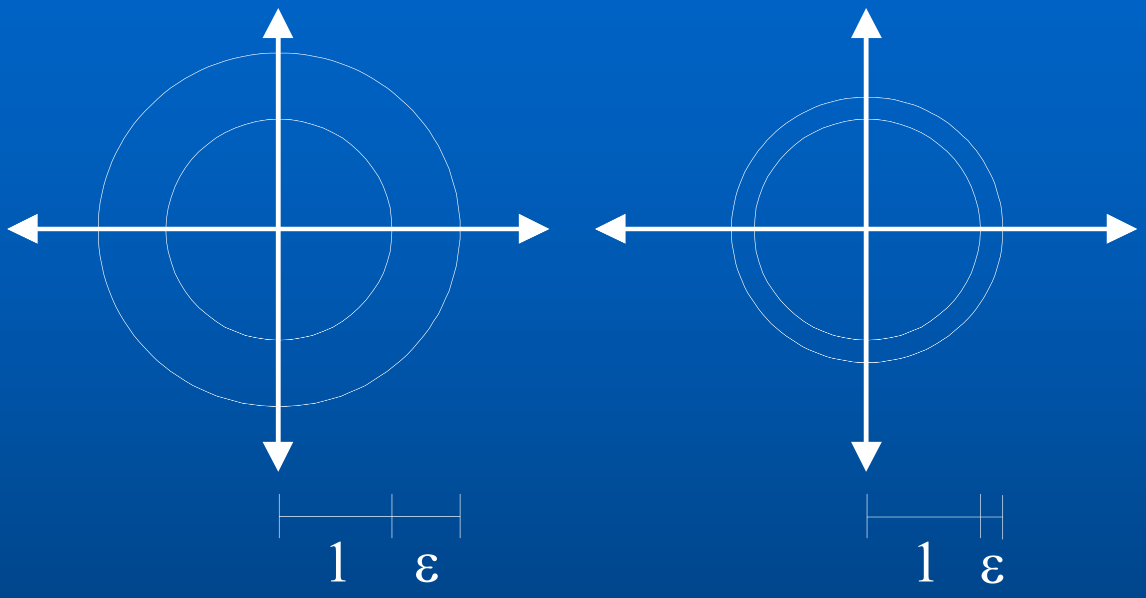
- Strengths of this overlap test:
  - 89 to 252 arithmetic operations per box overlap test
  - Simple guard against arithmetic error
  - No special cases for parallel/coincident faces, edges, or vertices
  - No special cases for degenerate boxes
  - No conditioning problems
  - Good candidate for micro-coding

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

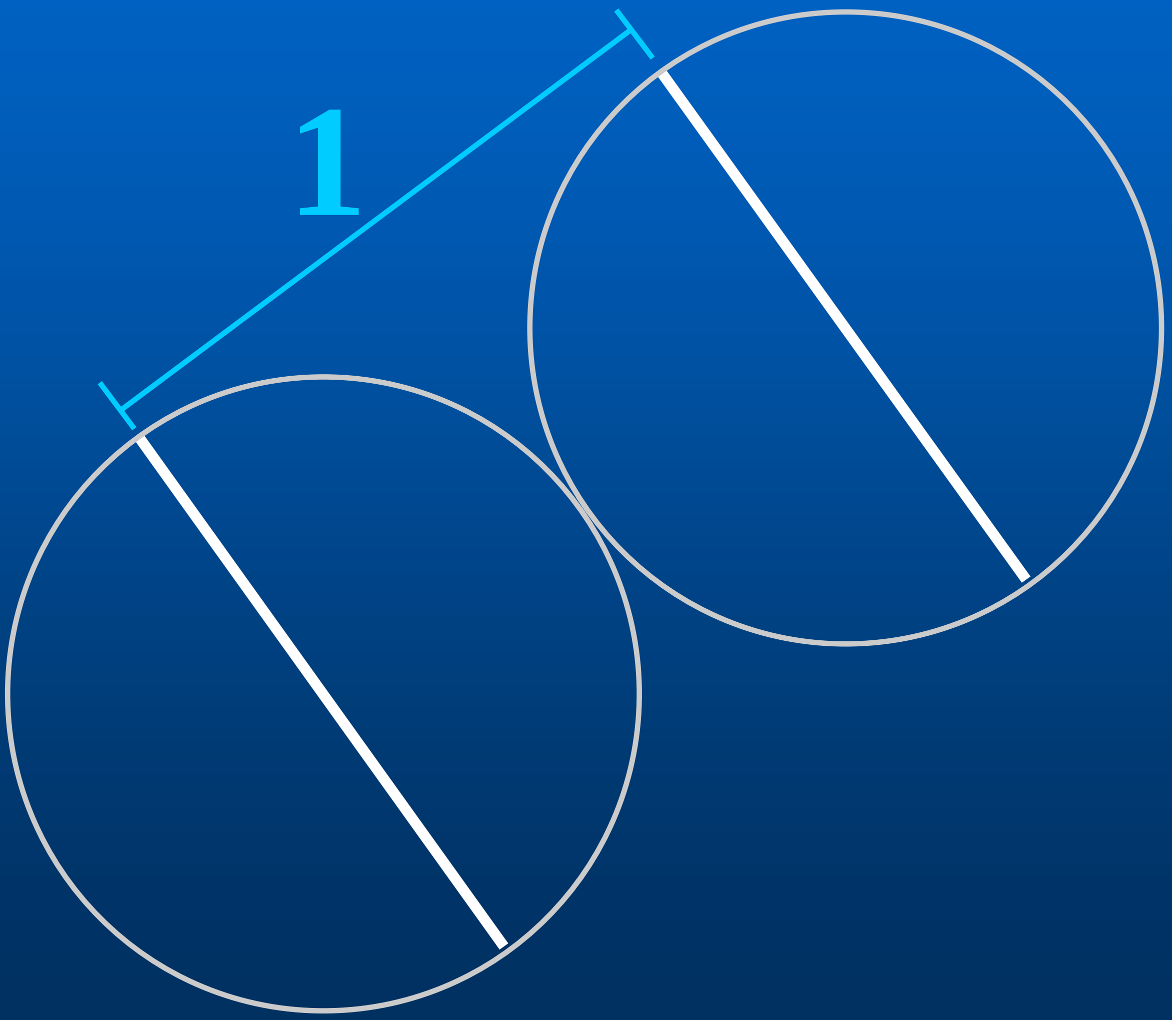
Parallel Close Proximity



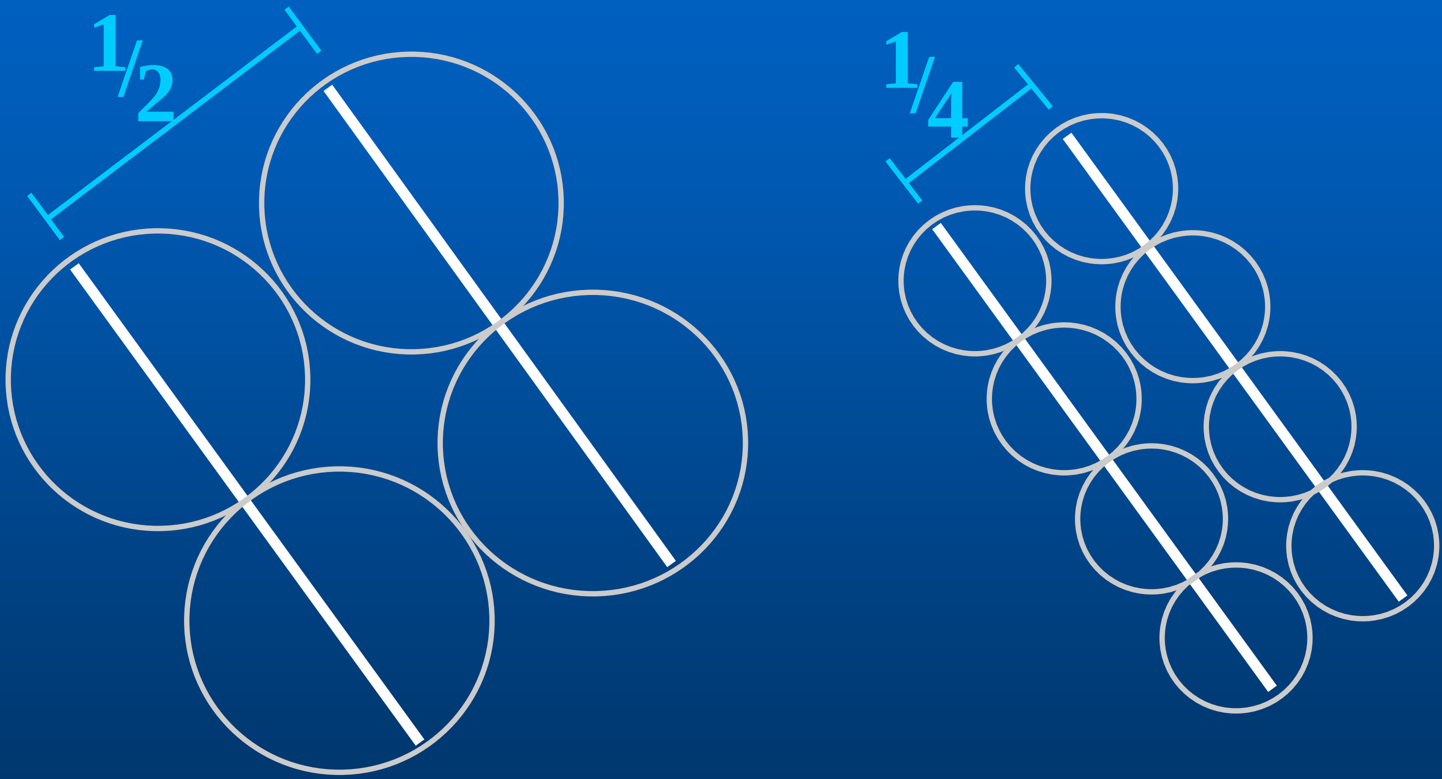
Two models are in *parallel close proximity* when every point on each model is a given fixed distance (ε) from the other model.

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

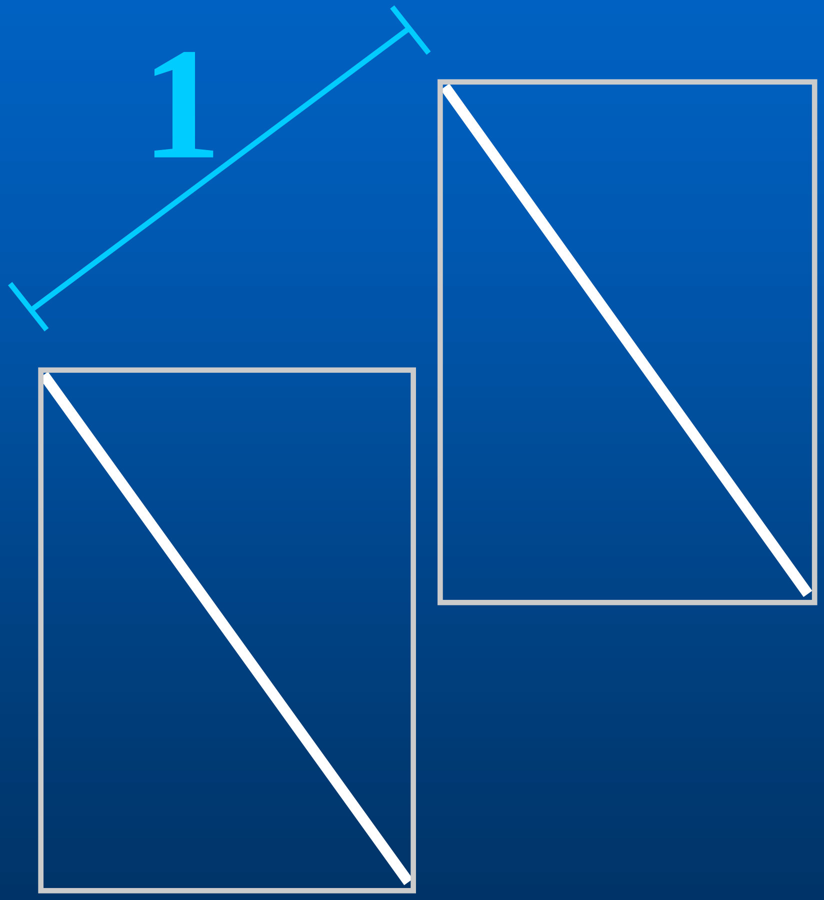
Parallel Close Proximity: Convergence



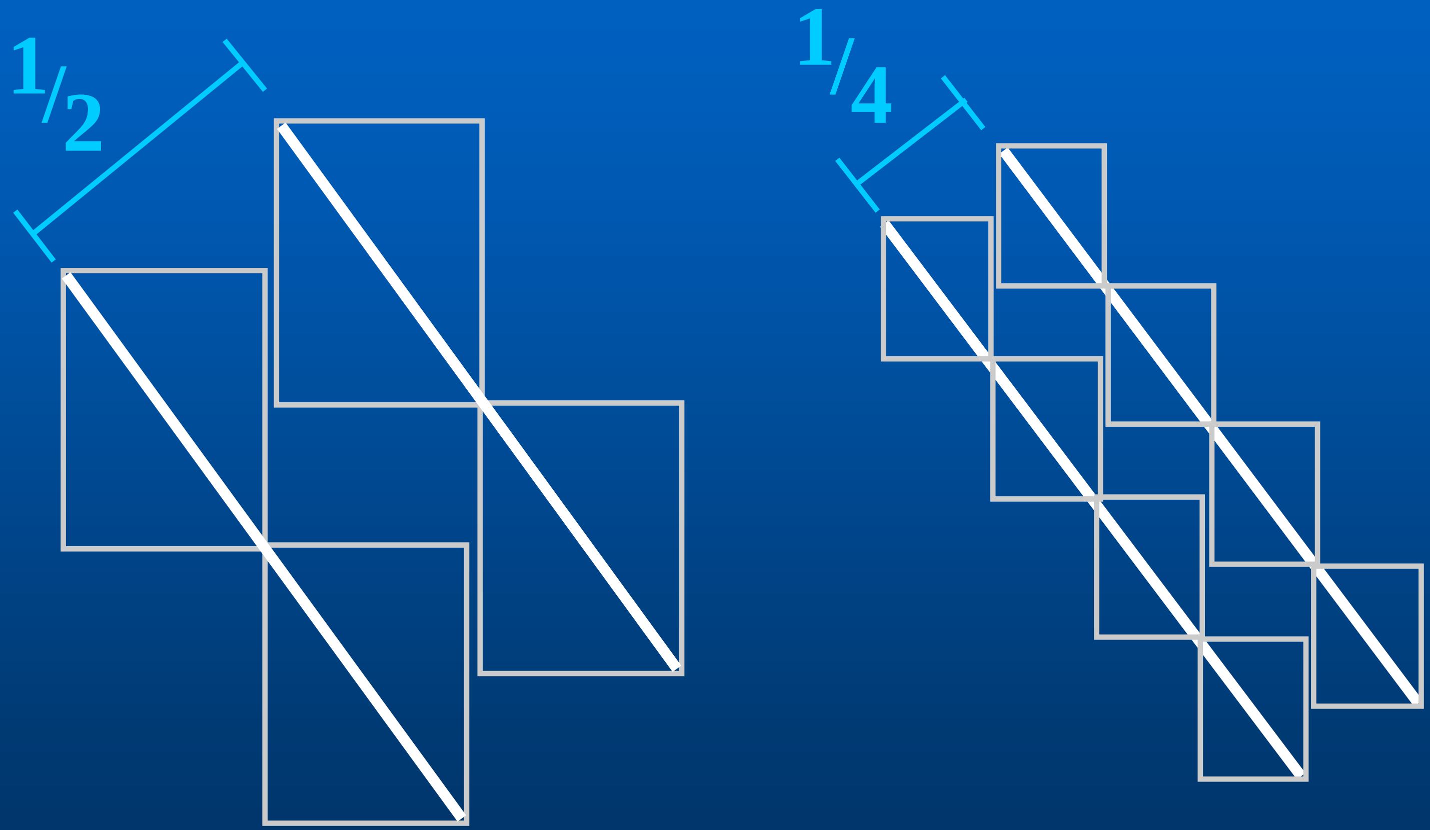
Parallel Close Proximity: Convergence



Parallel Close Proximity: Convergence

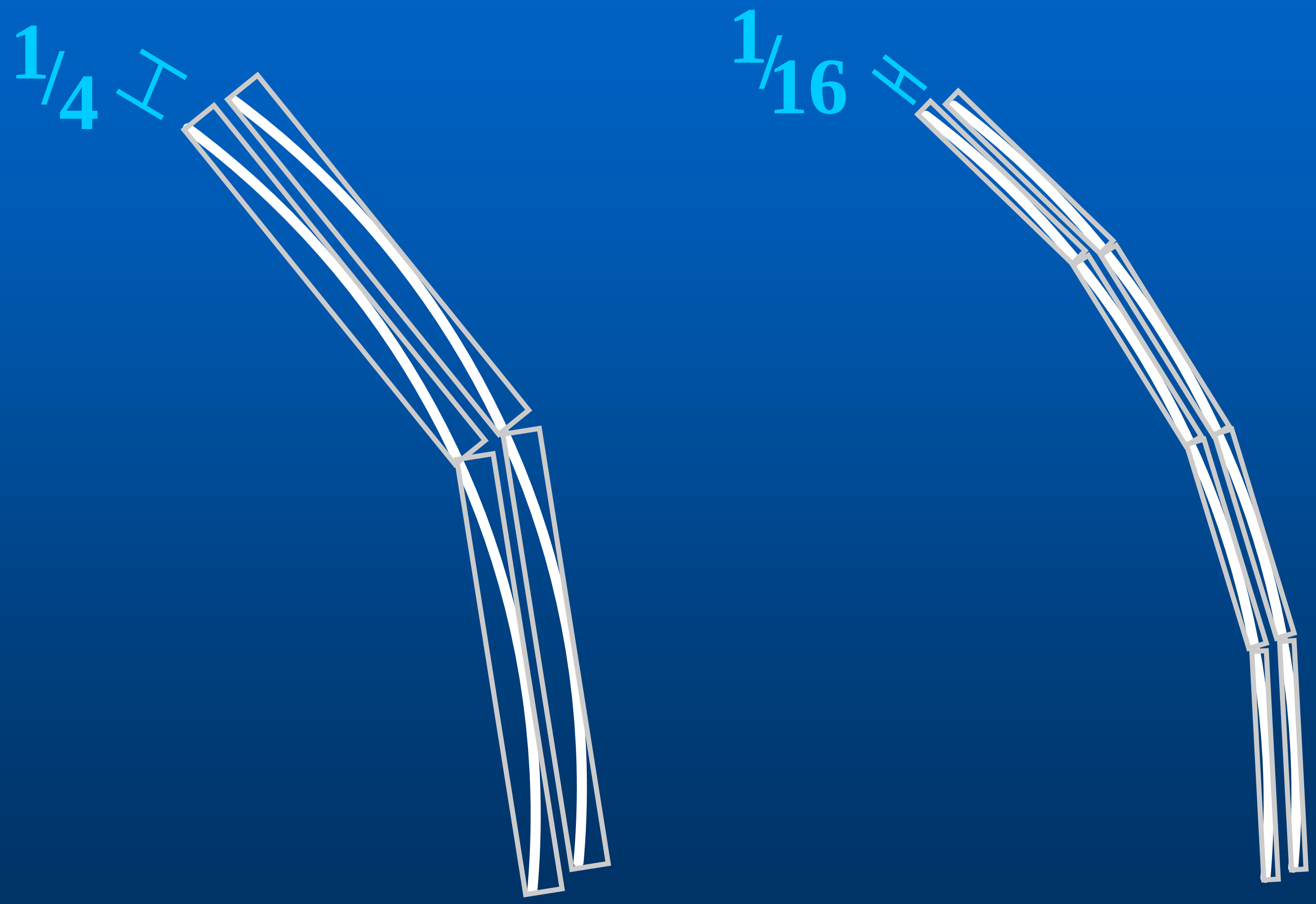


Parallel Close Proximity: Convergence

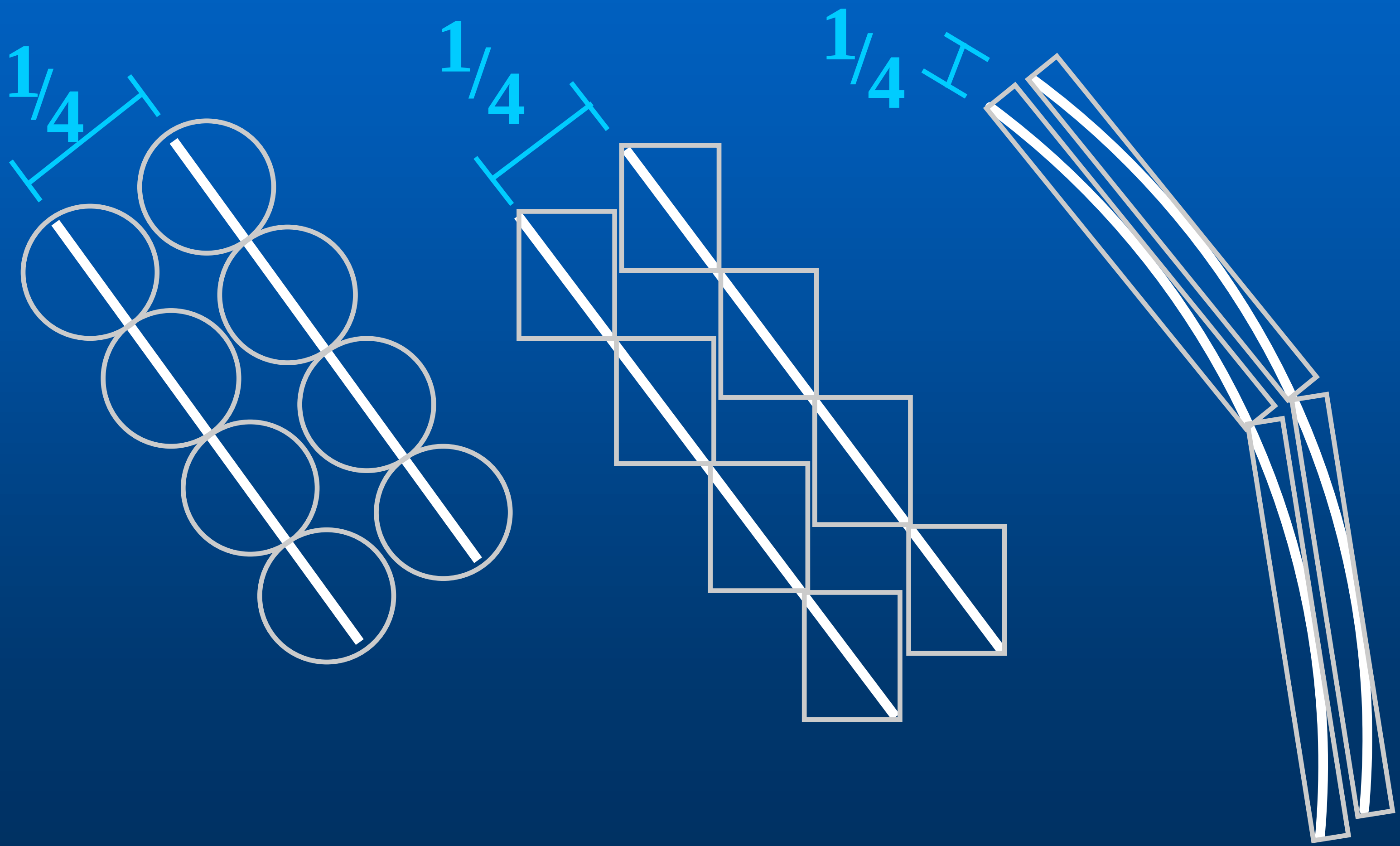


Parallel Close Proximity: Convergence

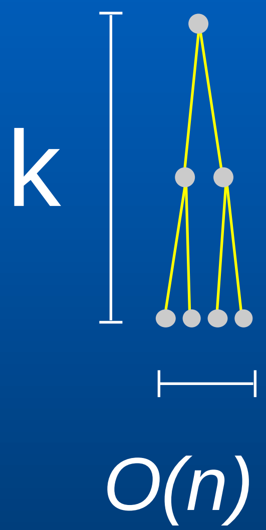
Parallel Close Proximity: Convergence

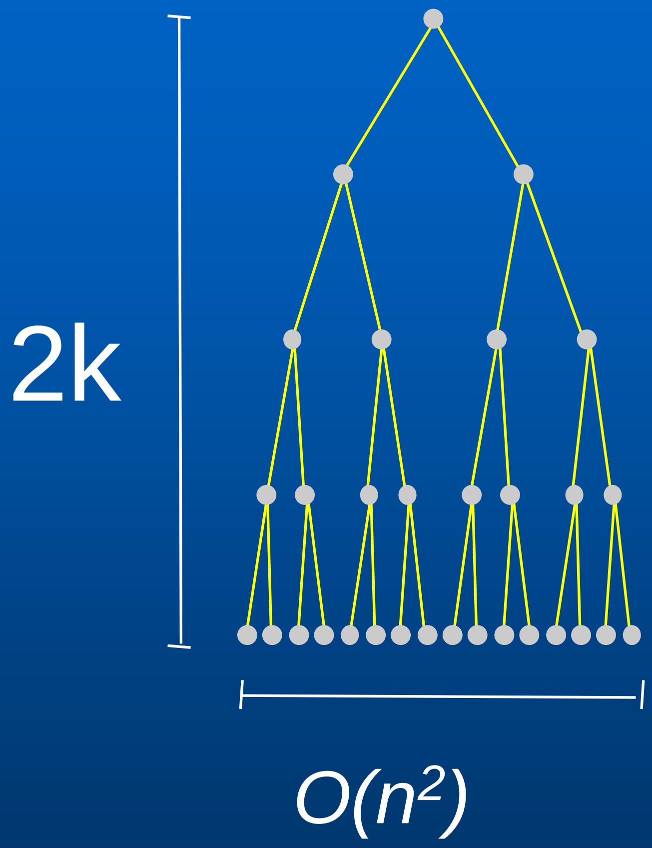


Parallel Close Proximity: Convergence



Performance: Overlap Tests

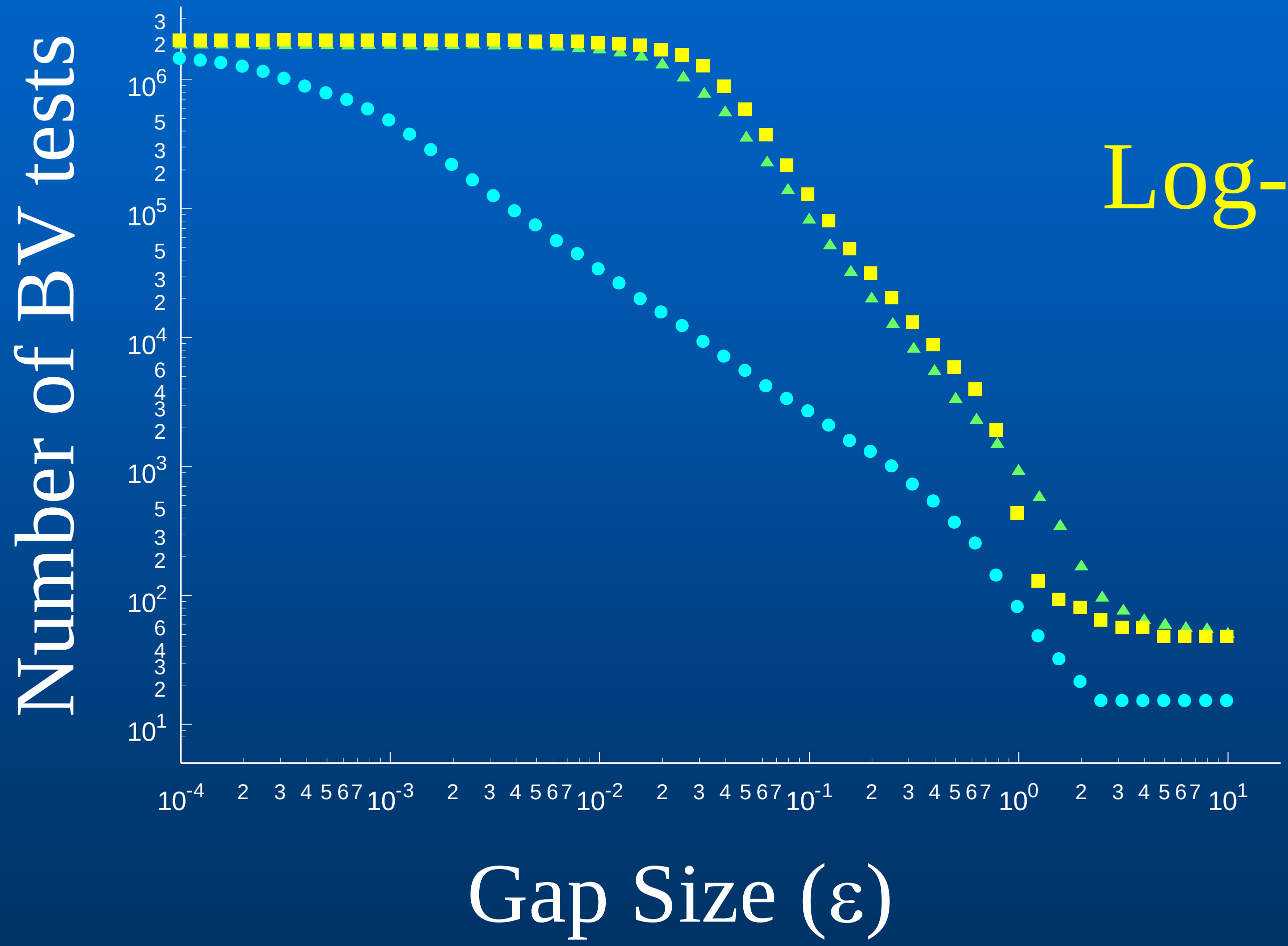




OBBs

Spheres & AABBs

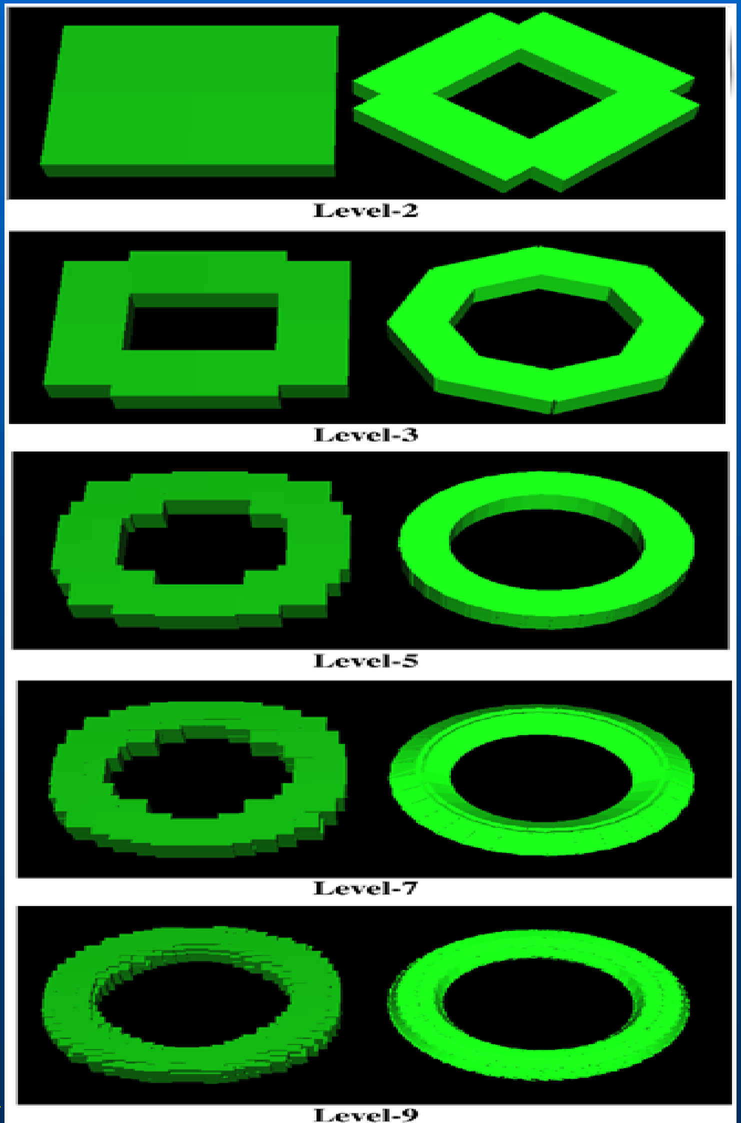
Parallel Close Proximity: Experiment



Log-log plot

OBBs asymptotically outperform AABBs and spheres

Example: AABB's vs. OBB's

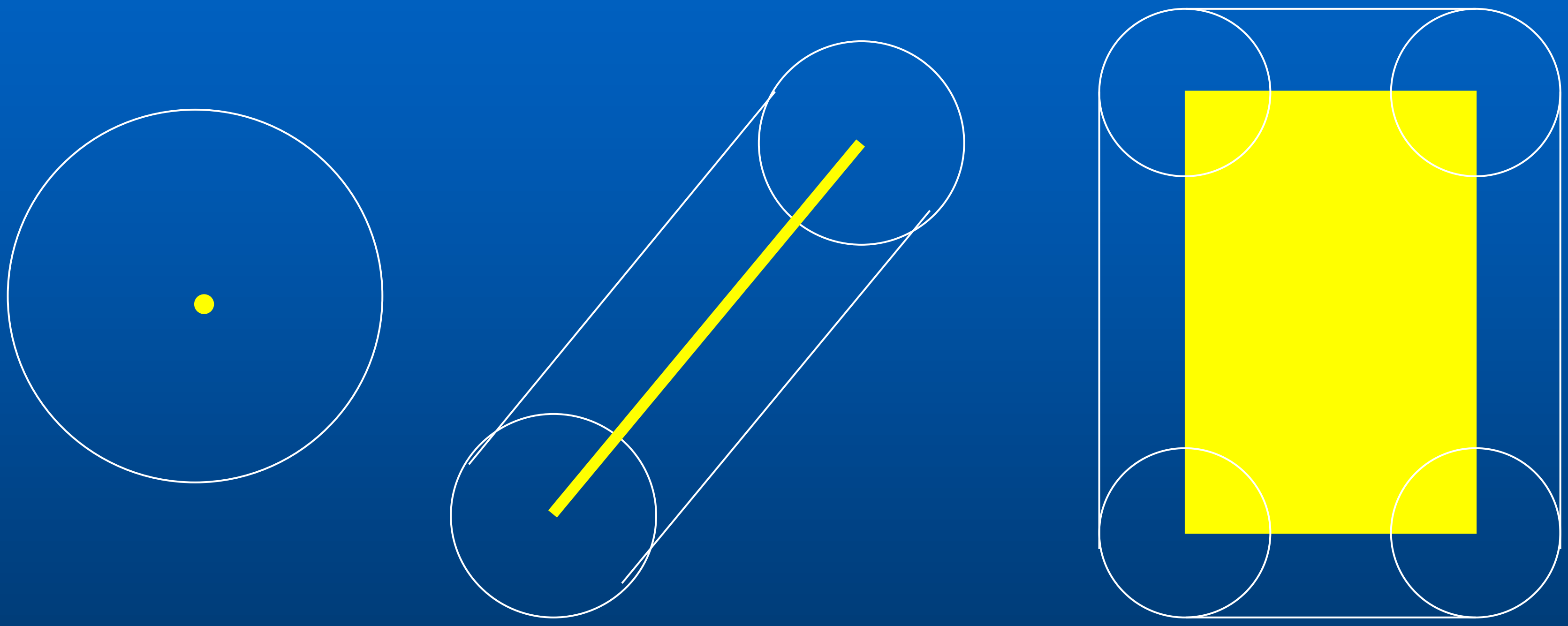


Approximation of a Torus

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

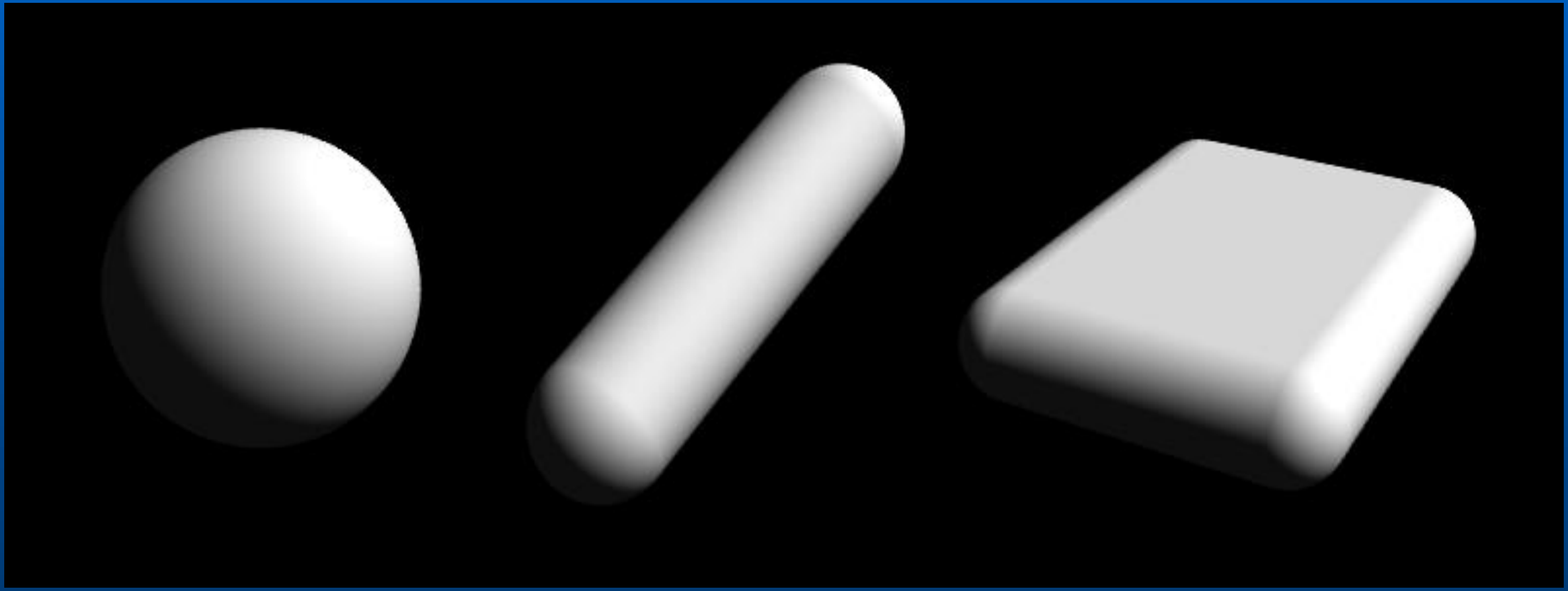
Hybrid Hierarchy of Swept Sphere Volumes



PSS LSS RSS

[LGLM99]

Swept Sphere Volumes (S-topes)



PSS **LSS** **RSS**

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

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.

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***

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/***