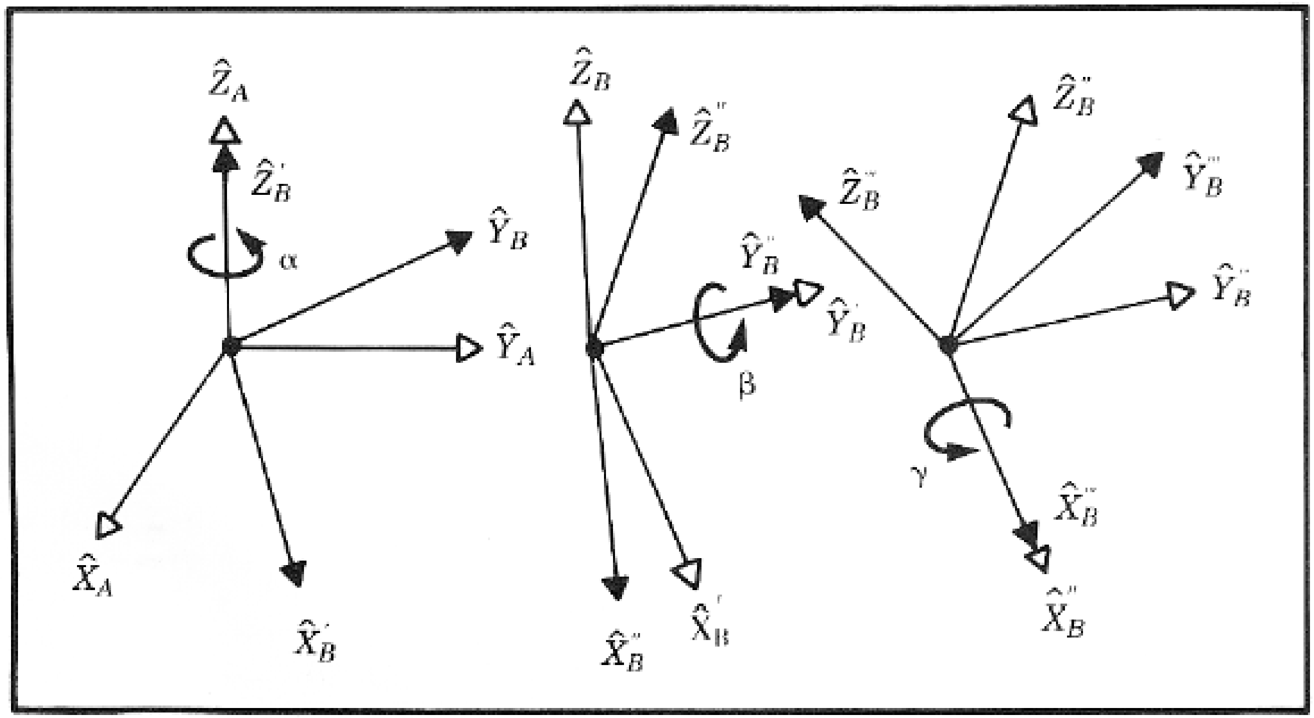
Collision Detection

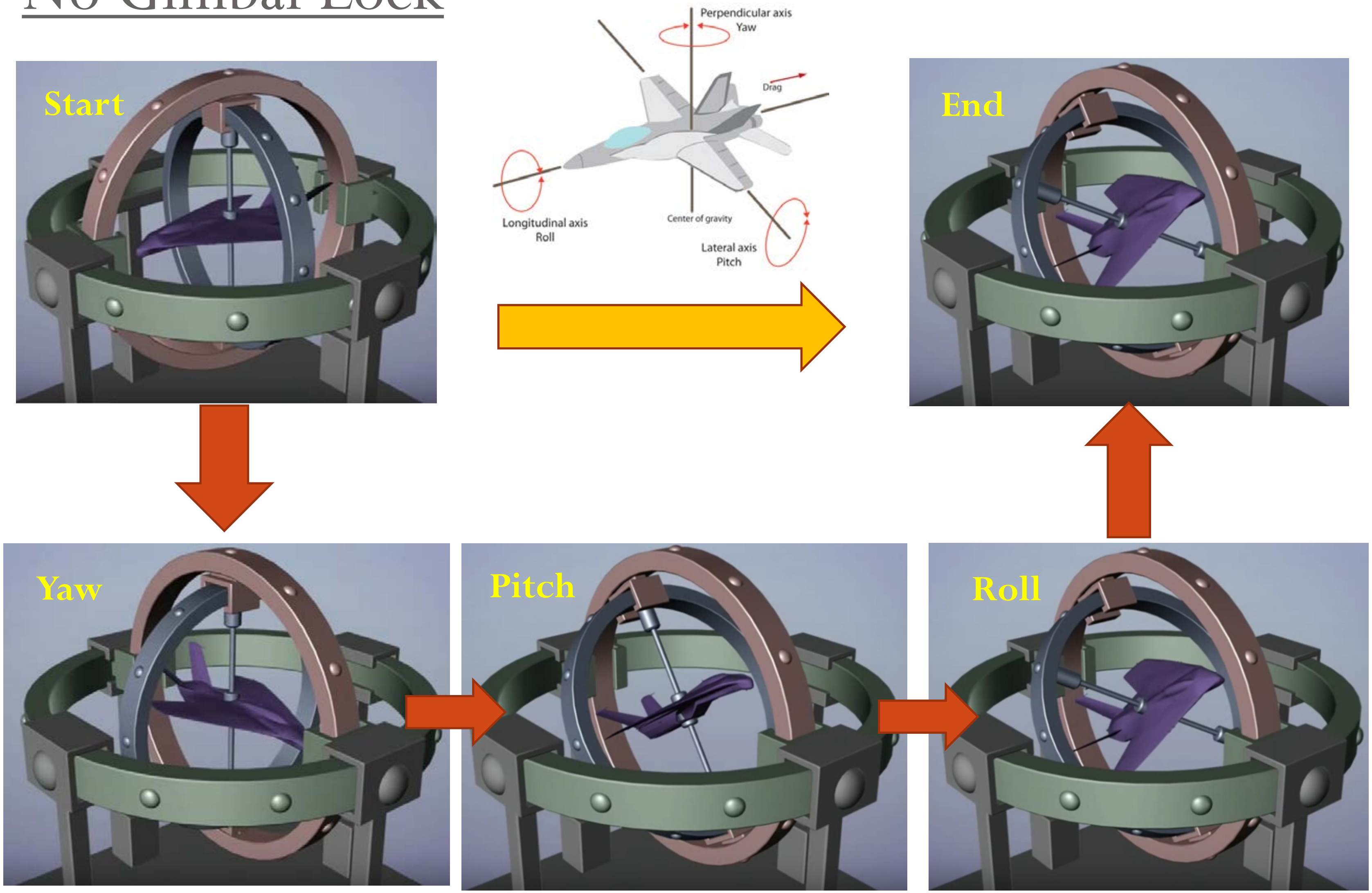
Jane Li
Assistant Professor
Mechanical Engineering & Robotics Engineering
http://users.wpi.edu/~zli11

Euler Angle

- Euler angle – Change the orientation of a rigid body to by
  - Applying sequential rotations about moving axes

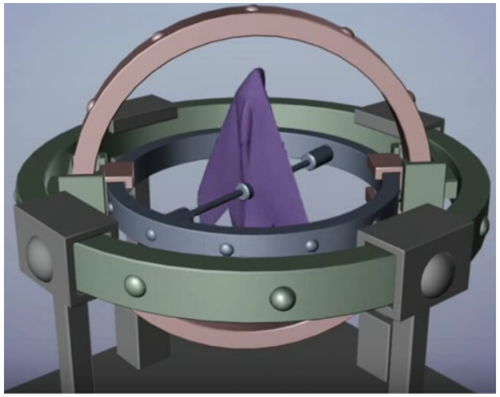


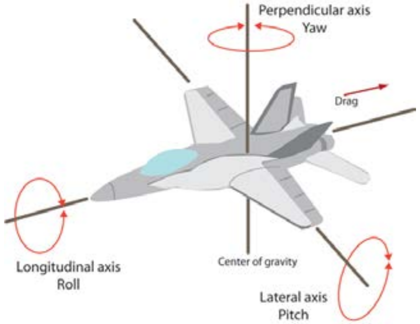
No Gimbal Lock



Euler Angle and Gimbal

- Applying Euler angle rotation **behaves** as if
  - Changing the orientation of an object using real gimbal set – a mechanism
  - As a mechanism, gimbal set can have **singularity**
- Gimbal lock
  - At this configuration, gimbal set can change the roll in many ways, but
    - Cannot change the yaw of the plane, without changing the pitch at the same time
    - → Lose the control of one DOF for **yaw**





Gimbal Lock – Singularity Problem

- Singularities
  - Why does the ring for yaw and ring for roll do the same thing?
- Let's say this is our convention:

$$R = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta \\ 0 & \sin\beta & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Lets set $\beta = 0$

$$R = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Multiplying through, we get:

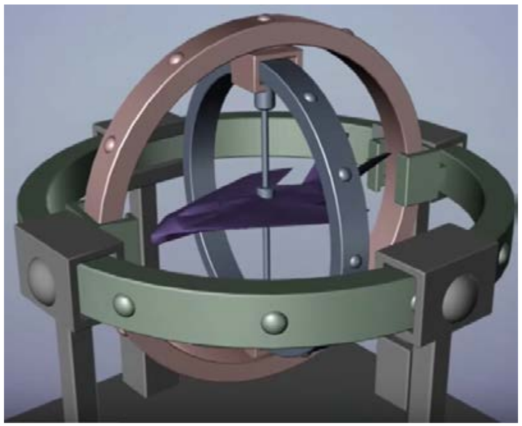
$$R = \begin{bmatrix} \cos\alpha\cos\gamma - \sin\alpha\sin\gamma & -\cos\alpha\sin\gamma - \sin\alpha\cos\gamma & 0 \\ \sin\alpha\cos\gamma + \cos\alpha\sin\gamma & -\sin\alpha\sin\gamma + \cos\alpha\cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

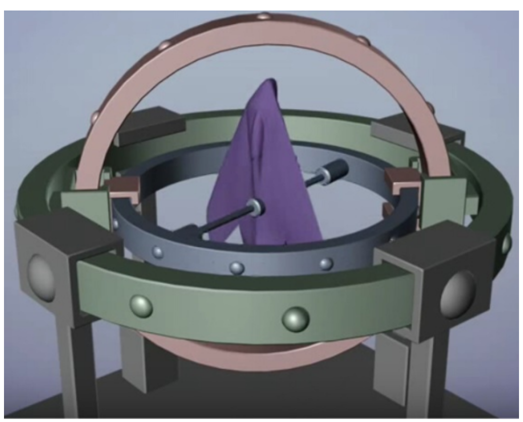
- Simplify:

$$R = \begin{bmatrix} \cos(\alpha+\gamma) & -\sin(\alpha+\gamma) & 0 \\ \sin(\alpha+\gamma) & \cos(\alpha+\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

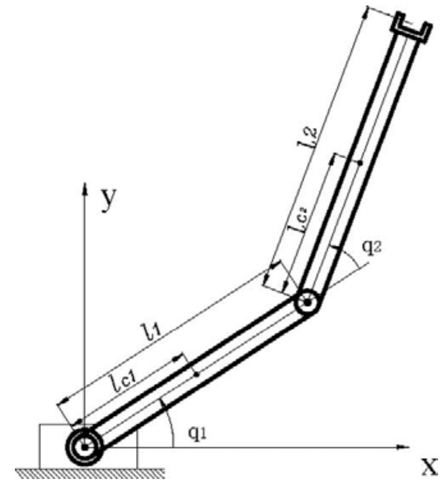
α and γ do the same thing!
We have lost a degree of freedom!

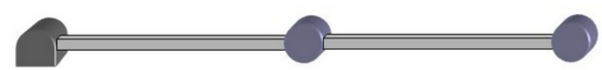
Gimbal Lock is a Singularity Problem



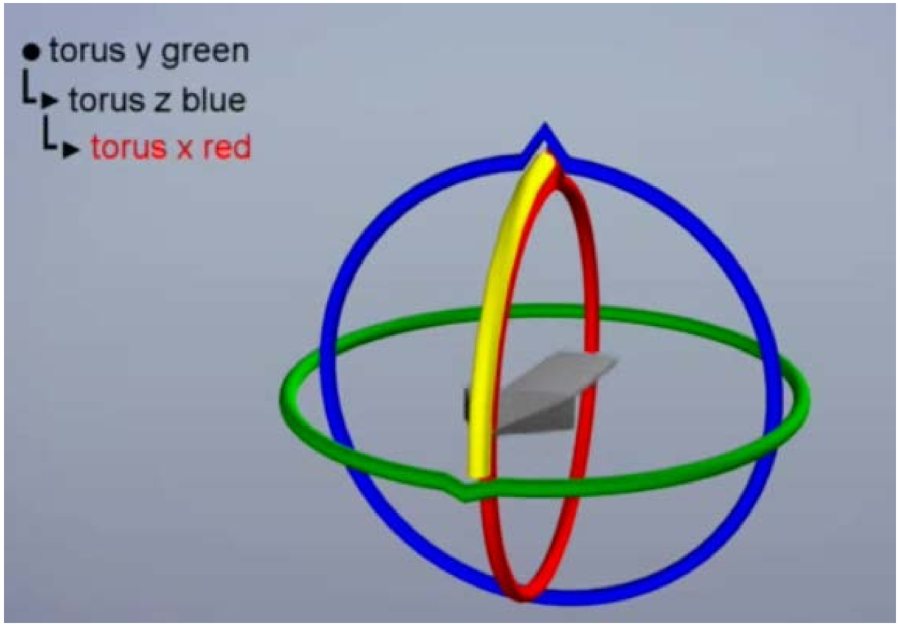


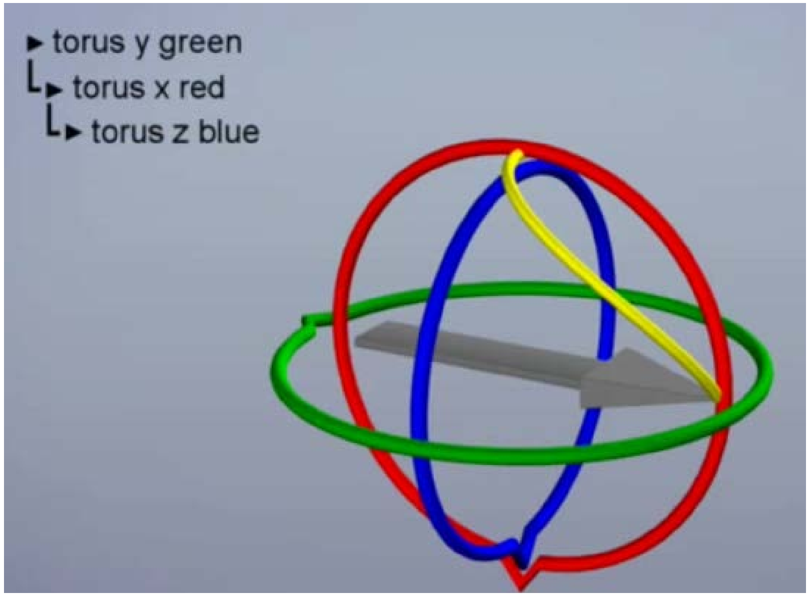
Lose the control of Yaw





Get out of gimbal lock?





To Pitch, First need to get out of gimbal lock
→Rotate ring for yaw back and forth, while rotate the ring for pitch
→Unexpected curved Motion

Why Rotation Matrix and Quaternion

- Why rotation matrix and quaternion do not have gimbal lock?
  - 3D Rotation → quaternion, rotation matrix --- one to one
  - 3D Rotation → Euler angles --- not one-to-one
    - Sometimes, the dimension of the Euler angle space drops to 2

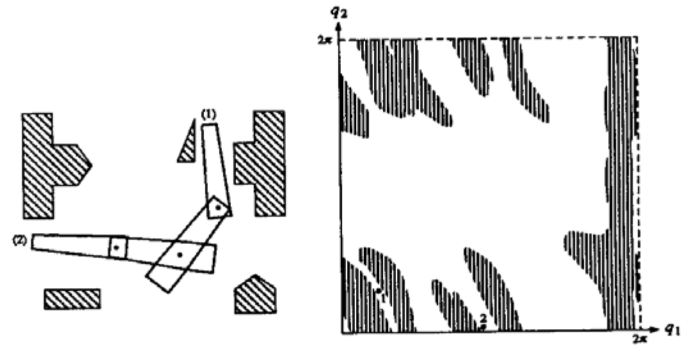
$$ {}^{A}_{B}R_{Z'Y'X'}(\alpha,\beta,\gamma) = R_Z(\alpha)R_Y(\beta)R_X(\gamma) = \begin{bmatrix} c\alpha & -s\alpha & 0 \\ s\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\beta & 0 & s\beta \\ 0 & 1 & 0 \\ -s\beta & 0 & c\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\gamma & -s\gamma \\ 0 & s\gamma & c\gamma \end{bmatrix} $$

1 2 3

$$ {}^{A}_{B}R_{Z'Y'X'}(\alpha,\beta,\gamma) = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix} $$

Motivation

- Find a path in C-space



  - Compute C_{obs} – Hard
  - Check if a configuration is collision – Easy
- Collision detection
  - For a single configuration
  - Along a path/trajectory

Collision Detection

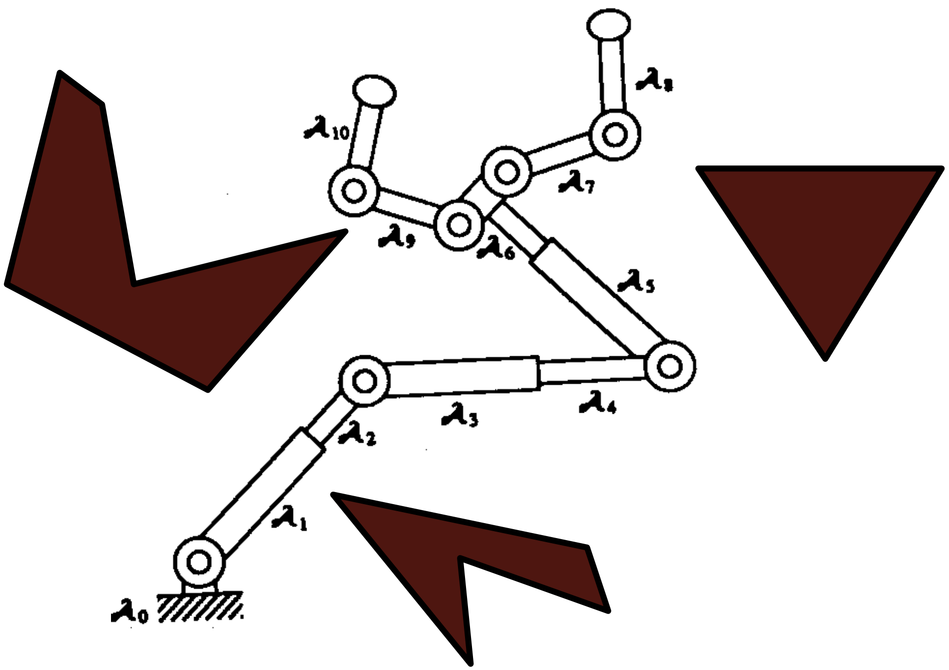
- Speed is **very** important
  - Need to check collision for **large number of** configurations
  - For most planners, runtime for real-world task depends **heavily** on the speed of collision checking
- Tradeoff
  - Speed
  - Accuracy
  - Memory usage
  - Increase speed → more memory, less accuracy

Crowd Simulation

Figure from Kanyuk, Paul. "Brain Springs: Fast Physics for Large Crowds in WALLdr E." IEEE Computer Graphics and Applications 29.4 (2009).

Self-Collision Checking for Articulated Robot

- Self-collision is typically not an issue for mobile robots
- Articulated robots must avoid self-collision
  - Parent-child link – set proper joint angle limits
  - With root or other branches – e.g. Humanoid robot?



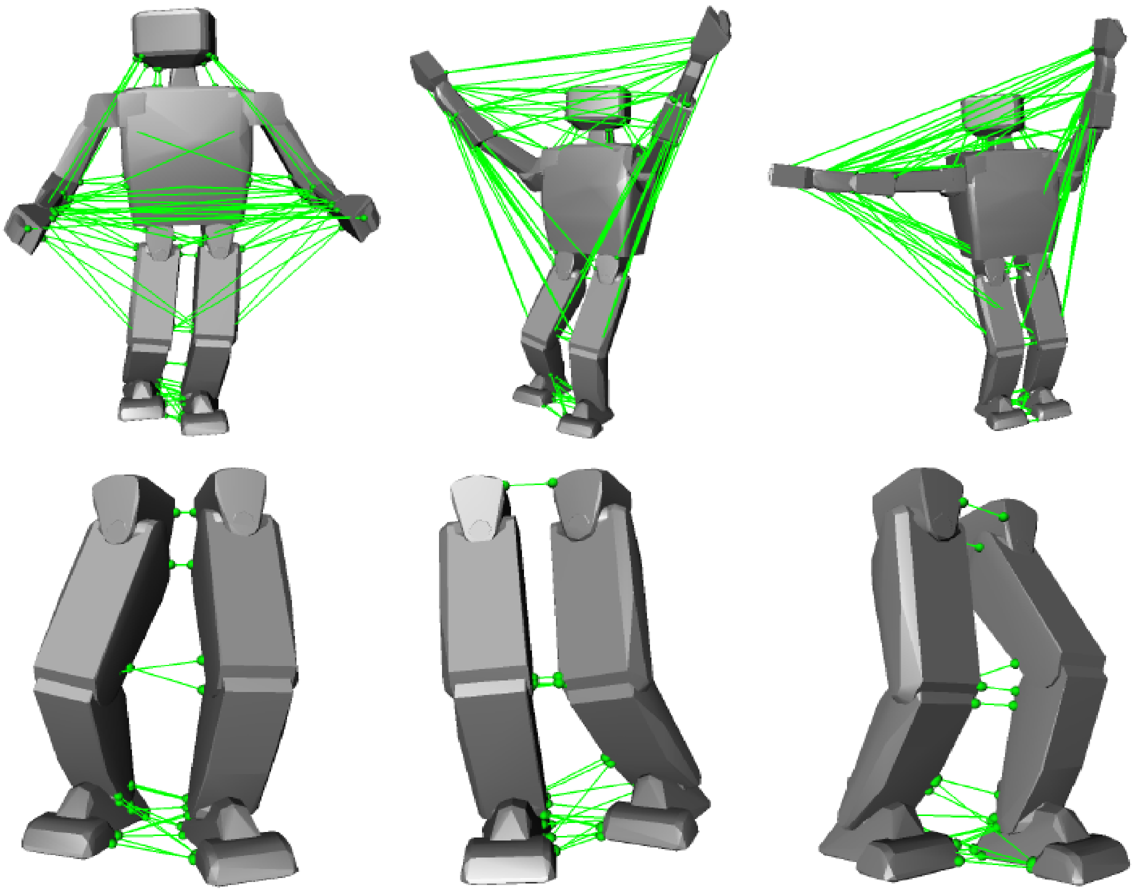
Self-Collision Checking For Humanoid Robot

$$P = \left(\sum_{i=1}^{N-1} i\right) - (N-1) = \sum_{i=1}^{N-2} i$$

$$P = \frac{N^2 - 3N + 2}{2}$$

For $N = 31$,

$$\boxed{P = 435.}$$



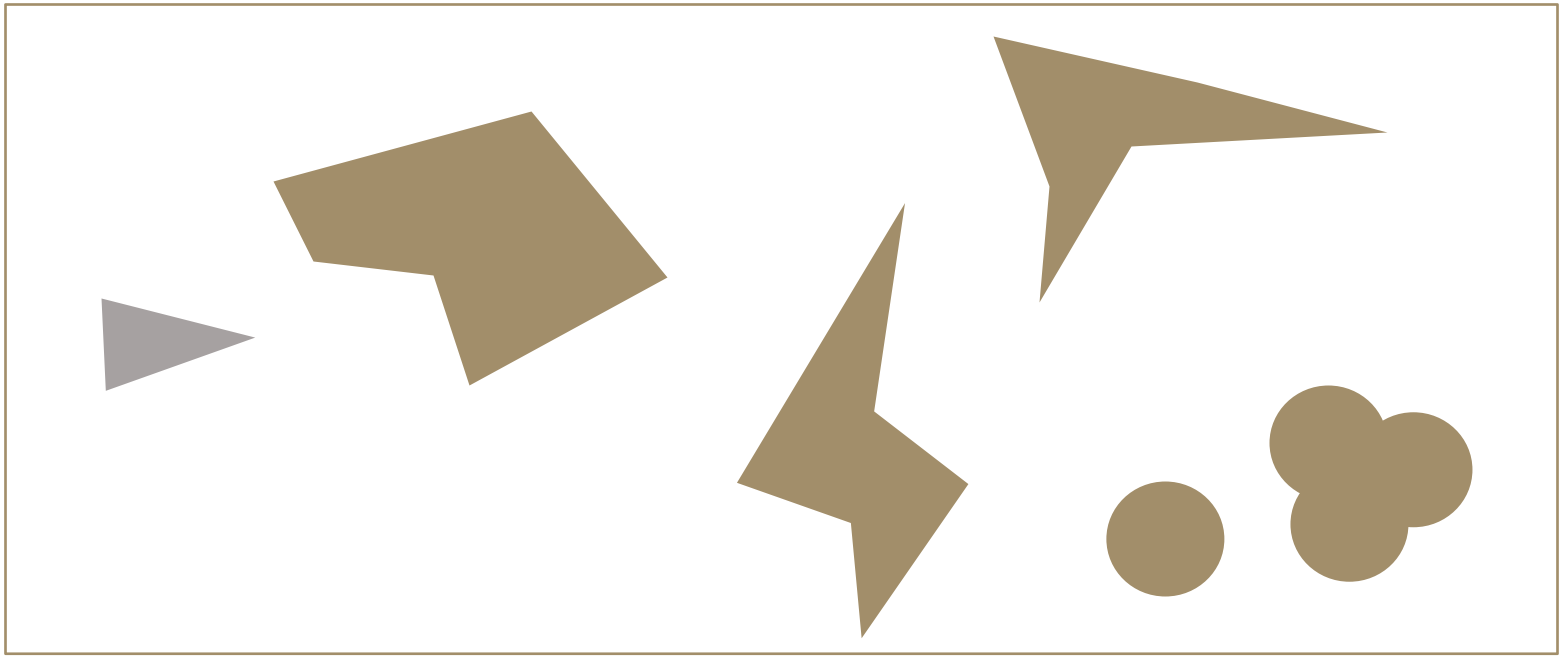
(J. Kuffner et al. Self-Collision and Prevention for Humanoid Robots. Proc. IEEE Int. Conf. on Robotics and Automation, 2002)

Outline

- Representing Geometry
- Methods
  - Bounding volumes
  - Bounding volume Hierarchy
- Dynamic collision detection
- Collision detection for Moving Objects
  - Feature tracking, swept-volume intersection, etc.

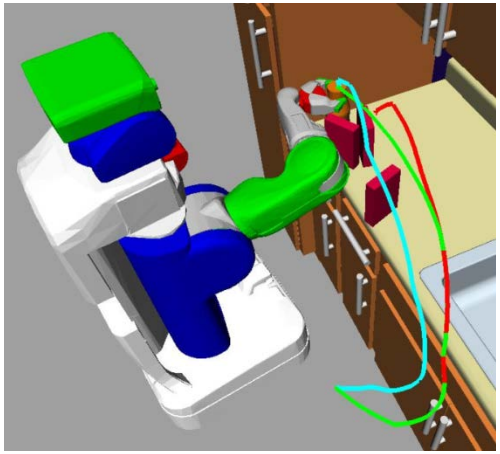
2D Representation

- 2D robots and obstacles are usually represented as
  - Polygons
  - Composites of discs

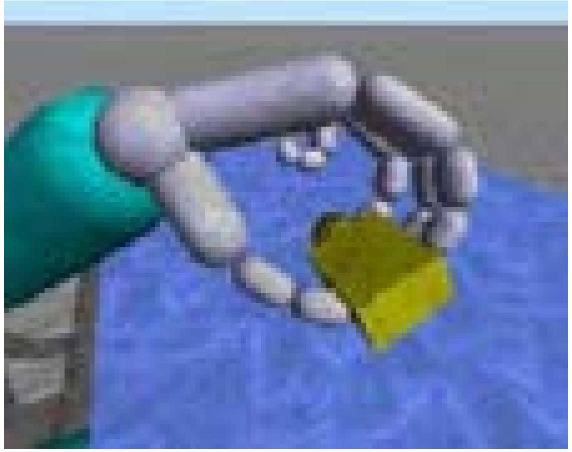


3D Representation

- Many representations - most popular for motion planning are
  - Triangle meshes
  - Composites of primitives (box, cylinder, sphere)
  - Voxel grids



Triangle meshes

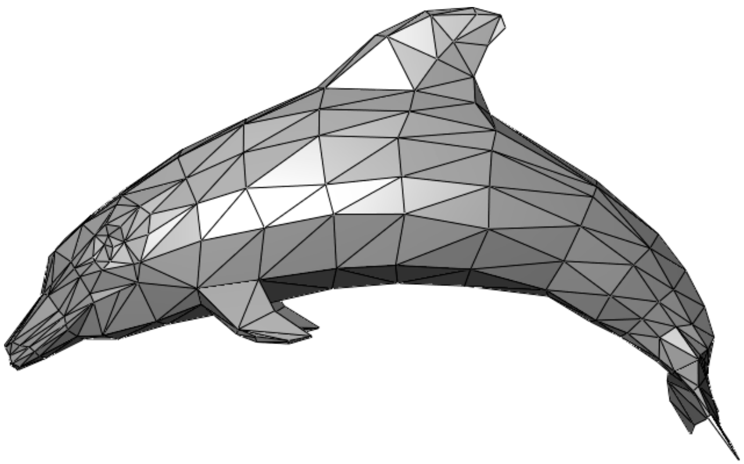


Composite of primitives

Voxel grid

3D Representation: Triangle Meshes

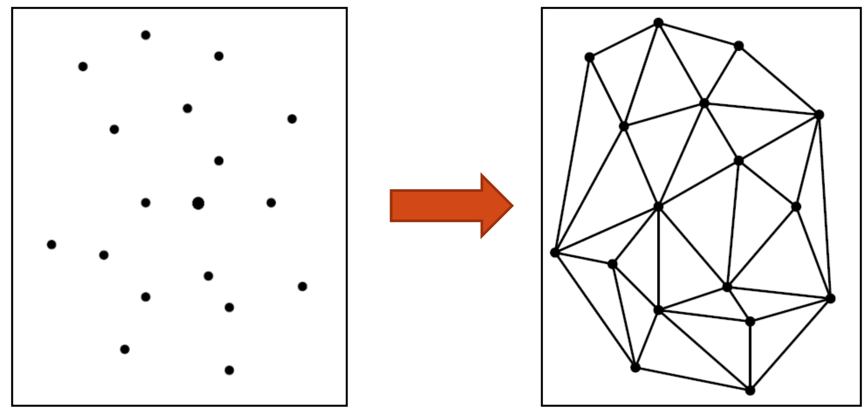
- Triangle mesh
  - A set of triangles in 3D that share common vertices and/or edges
- Most real-world shapes and be represented as triangle meshes

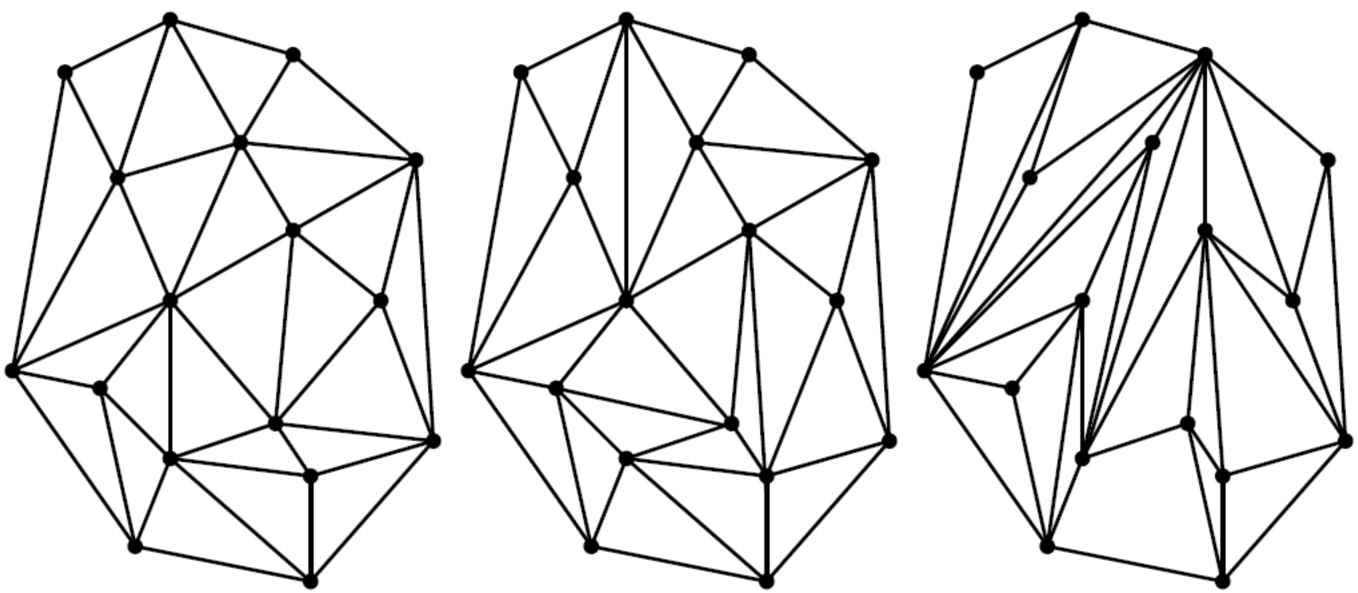


- Delaunay Triangulation
  - A good way to generate such meshes (there are several algorithms)

Delaunay Triangulation

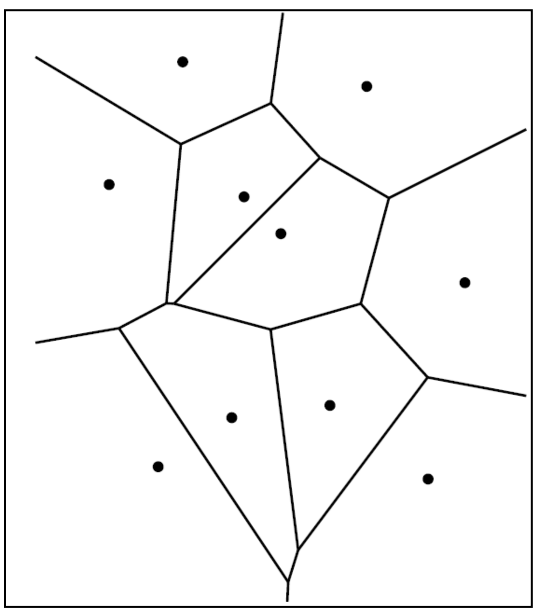
- Method
  - Sample on the terrain
  - Connect Sample points
  - Which triangulation?



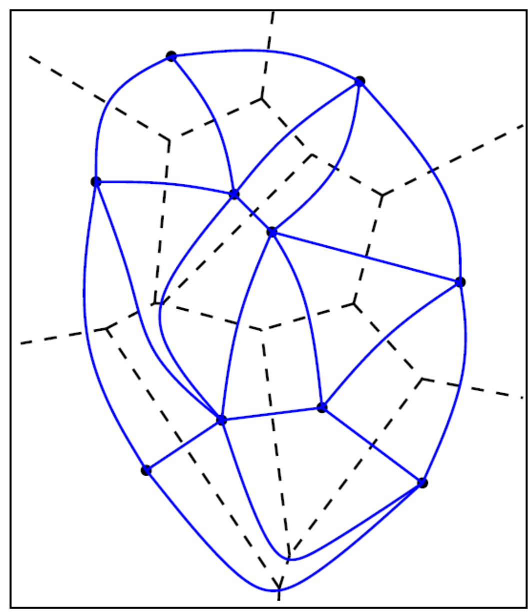


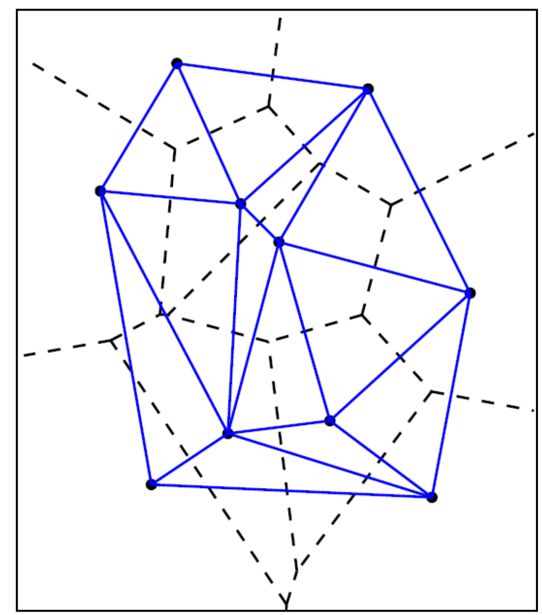
Delaunay Triangulation

- Goal – Avoid sliver triangle
  - Find the dual graph of Voronoi graph



Voronoi Graph

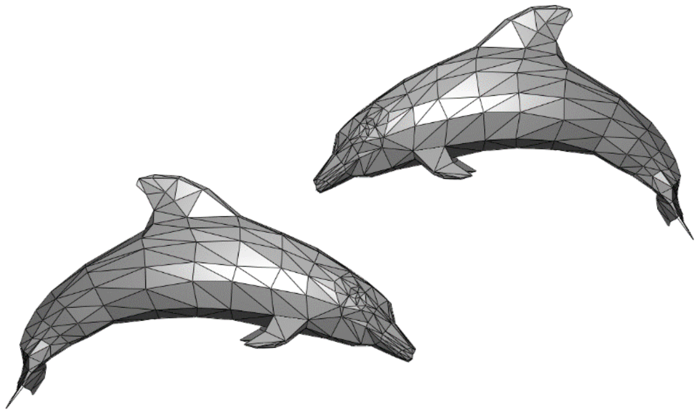




Delaunay Graph

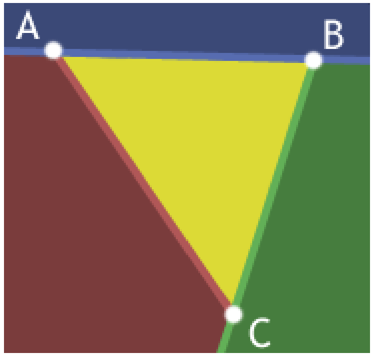
Collision Checking: Intersecting Triangle Meshes

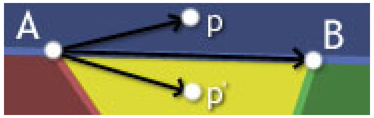
- The brute-force way
  - Check for intersection between every pair of triangles



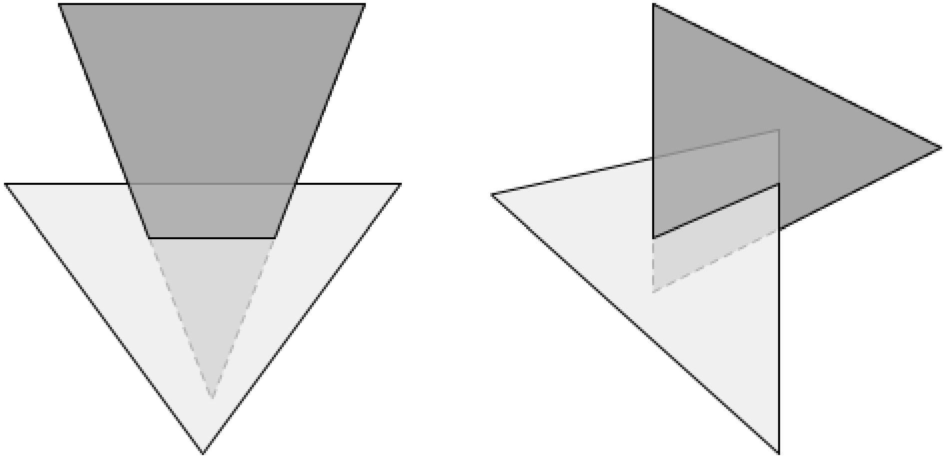
Collision Checking for Triangles

- Check if a point in a triangle



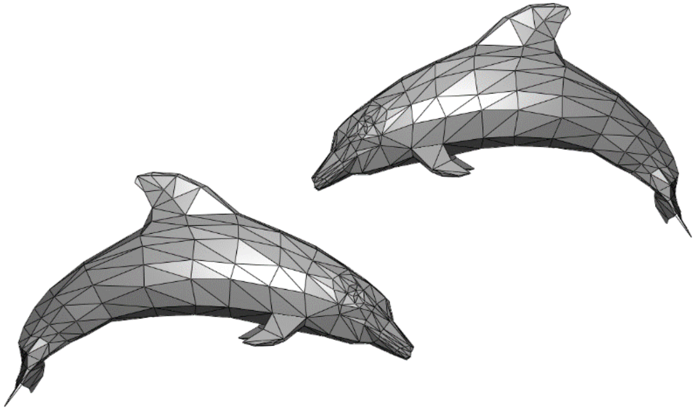


- Check if two triangles intersect



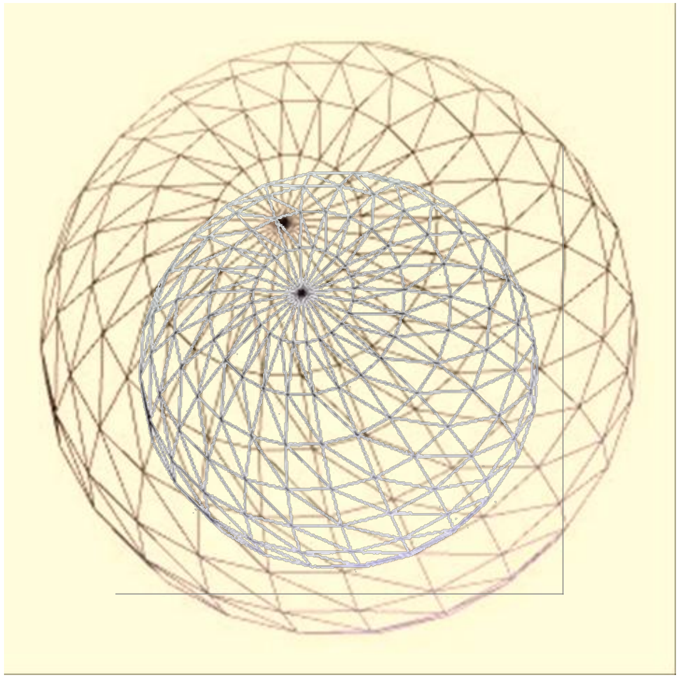
Collision Checking: Intersecting Triangle Meshes

- Triangle-Triangle intersection checks have a lot of corner cases; checking many intersections is slow
  - See "O'Rourke, Joseph. *Computational geometry in C*. Cambridge university press, 1994." for algorithms.
- Can we do better than all-pairs checking? – talk about it later …



3D Representation: Triangle Meshes

- Triangle Meshes are **hollow**!
- Be careful if you use them to represent solid bodies



One mesh inside another; no intersection

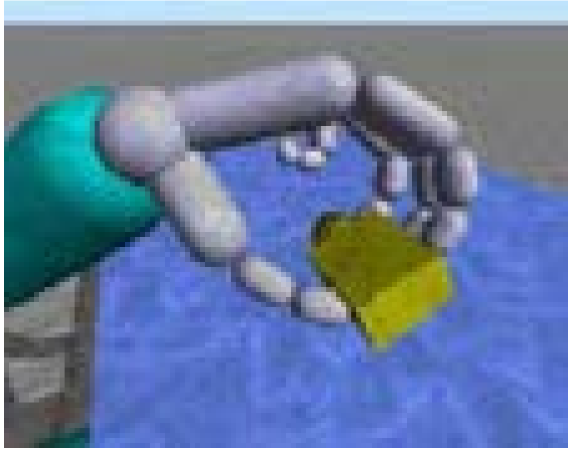
3D Representation: Triangle Meshes

- Complexity of collision checking increases with the number of triangles
  - Try to keep the number of triangles low

- Algorithms to simplify meshes exist but performance depends on shape

3D Representation: Composites of Primitives

- Advantages:
  - Can usually define these by hand
  - Collision checking is much faster/easier
  - Much better for simulation
- Disadvantages
  - Hard to represent complex shapes
  - Usually conservative (i.e. overestimate true geometry)

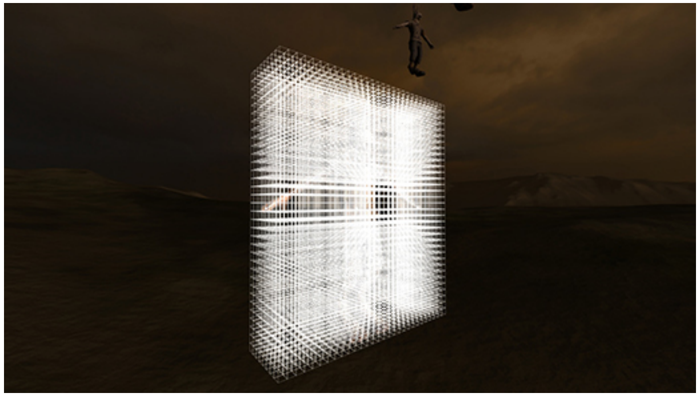


3D Representations: Voxel Grids

- Voxel –
  - Short for "volume pixel"
  - A single cube in a 3D lattice
- Not hollow like triangle meshes
  - Good for 'deep' physical simulations such as heat diffusion, fracture, and soft physics

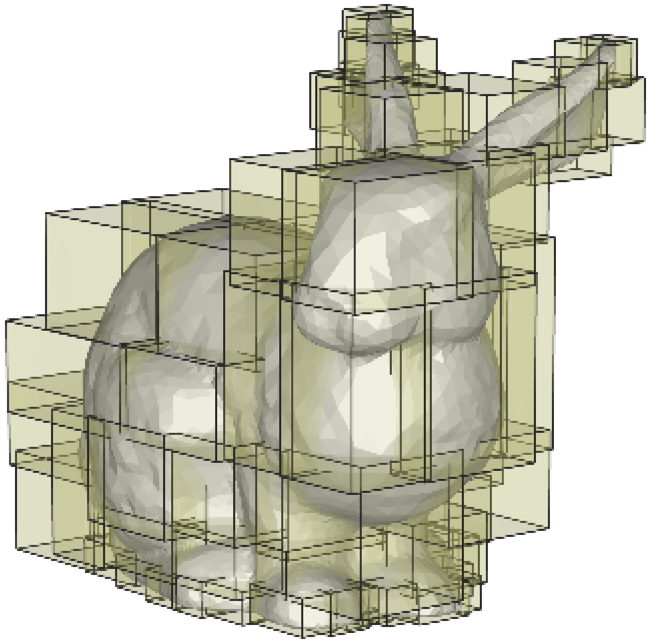
3D Representations: Voxel Grids

- How to make a voxel model from triangle mesh?
  - Grid the space
    - Grid resolution – without losing important details
    - Grid dimension – just enough to cover the model – efficiency
  - Solidify a shell representing the surface
  - Fill it in using a scanline fill algorithm



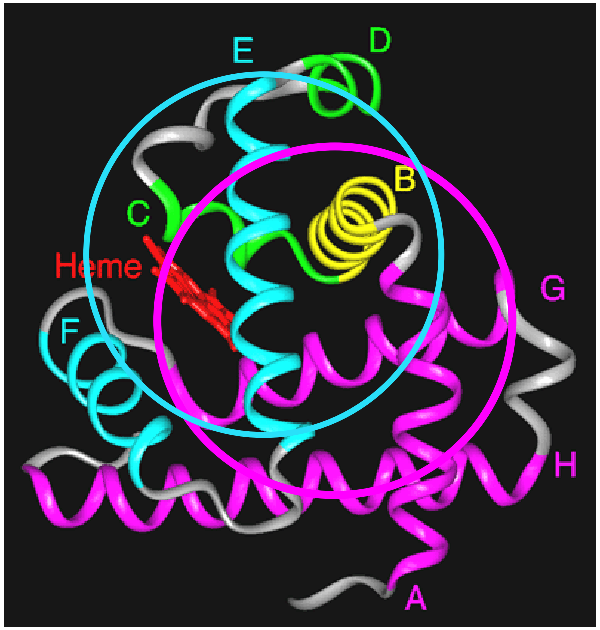
Bounding Volume

- Bounding Volume
  - A closed volume that completely contains the object (set).
  - If we don't care about getting the *true* collision,
    - Bounding volumes represents the geometry (conservatively)
- Various Bounding Volumes
  - Sphere
  - Axis-Aligned Bounding Boxes (AABBs)
  - Oriented Bound Boxes (OBBs)



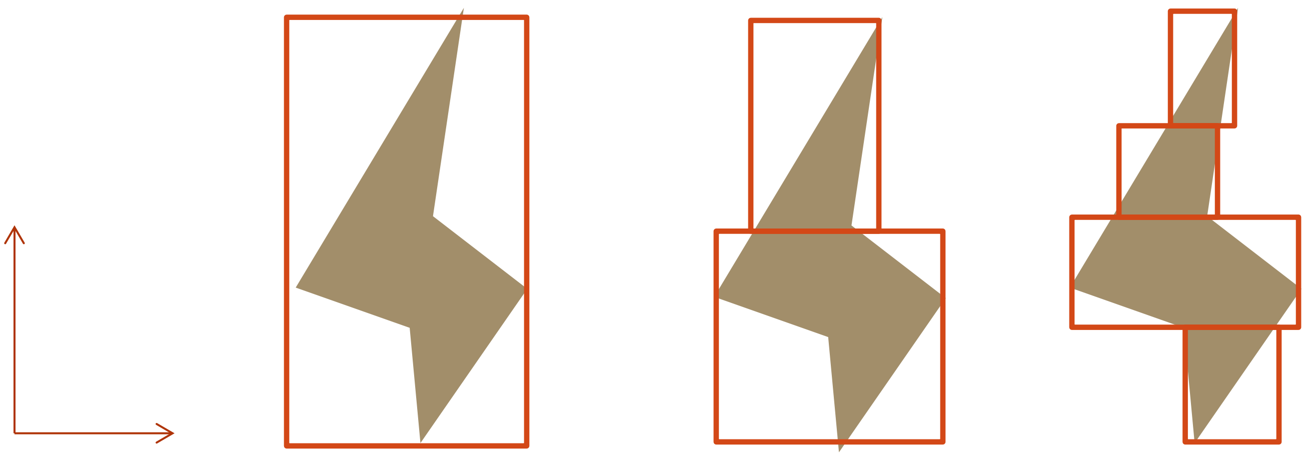
Spheres

- Invariant to rotation and translations,
  - Do not require being updated
- Efficient
  - constructions and interference tests
- Tight?



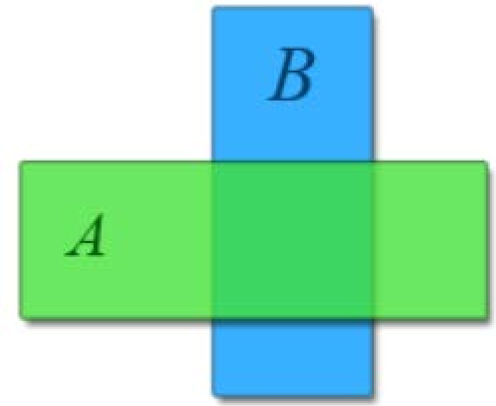
AABBs

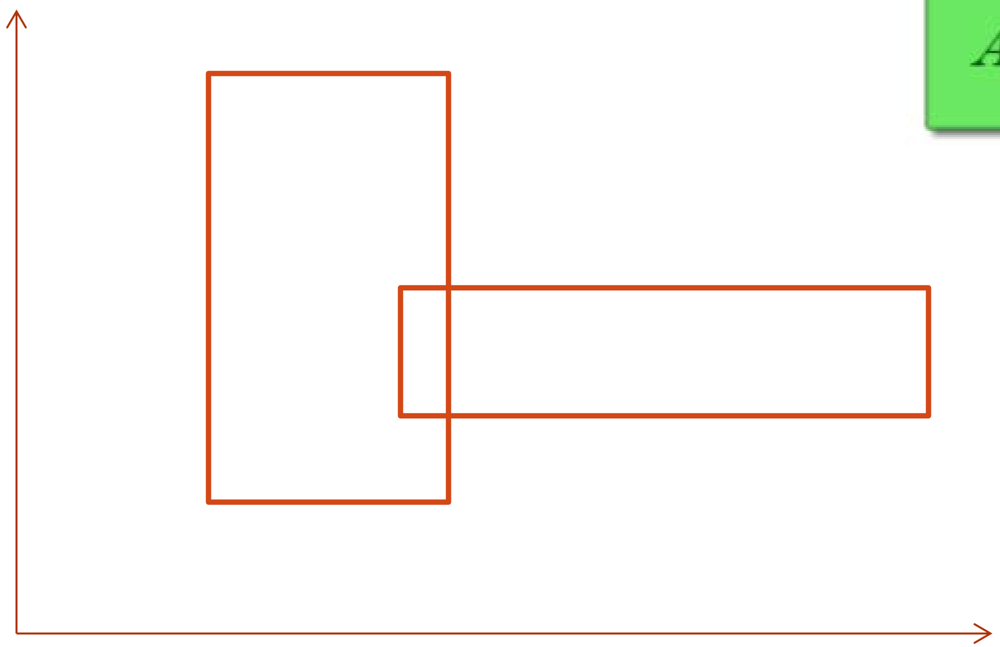
- Axis-Aligned Bounding Boxes (AABBs)
  - Bound object with one or more boxes oriented along the same axis



AABBs

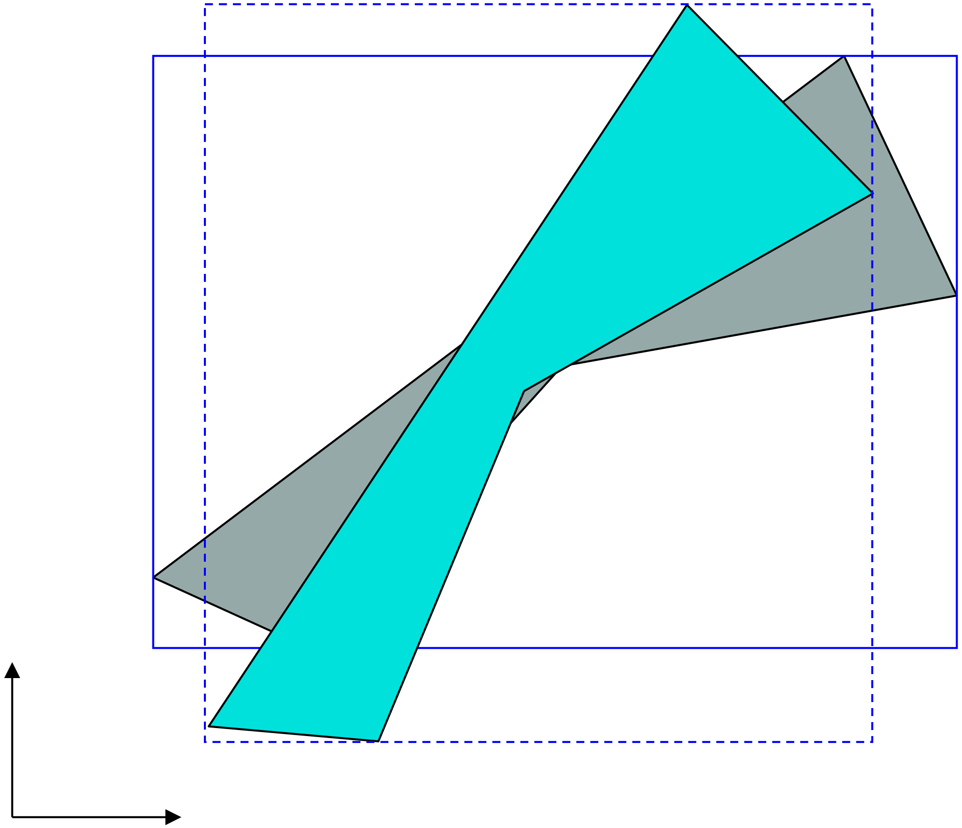
- How can you check for intersection of AABBs?





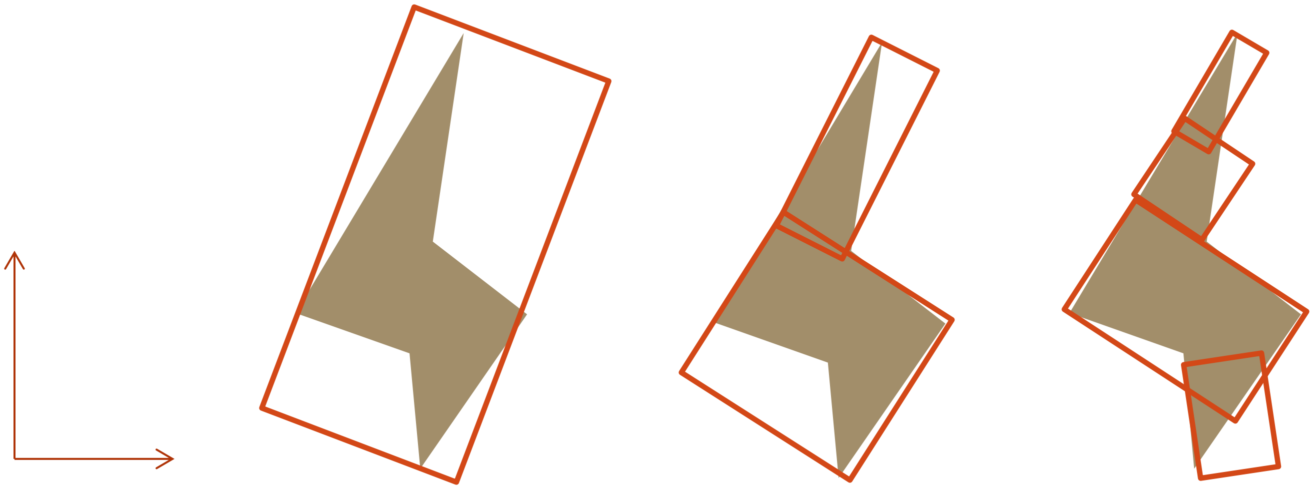
AABBs

- Not invariant
- Efficient
- Not tight



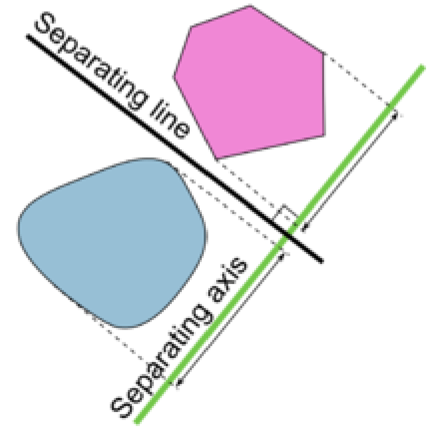
OBBs

- Oriented Bound Boxes (OBBs) are the same as AABBs except
  - The orientation of the box is not fixed



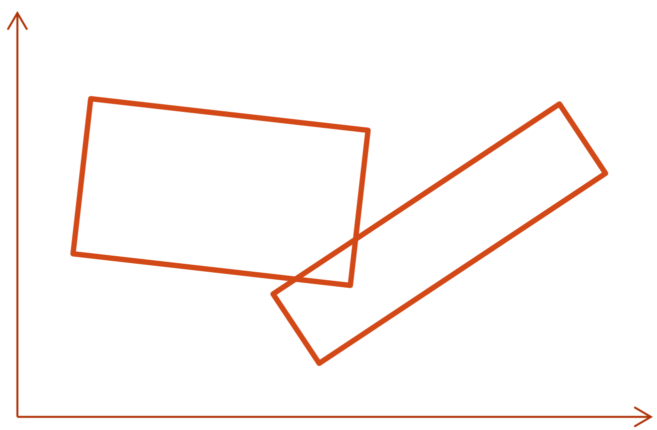
- OBBs can give you a tighter fit with fewer boxes

OBBs

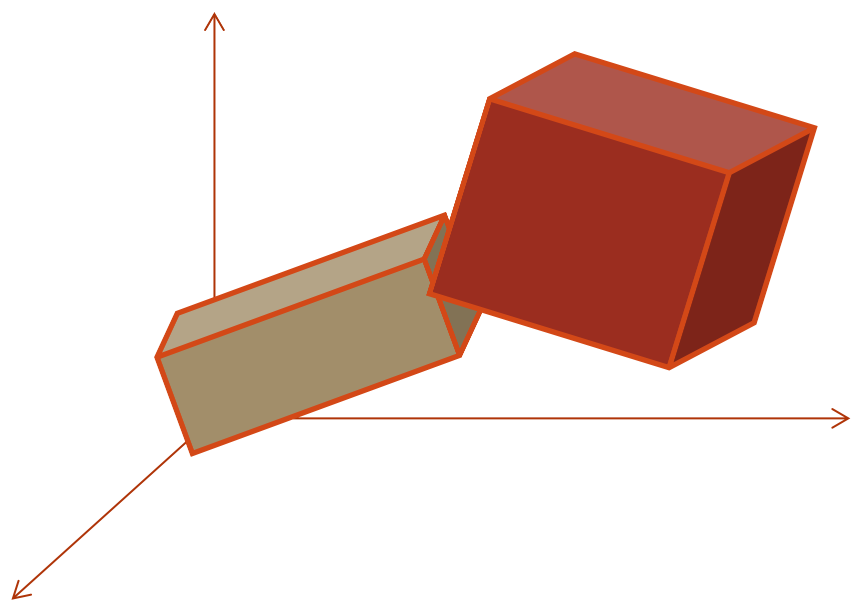


- How do you check for intersection of OBBs?
  - Hyperplane separation theorem

In 2D?

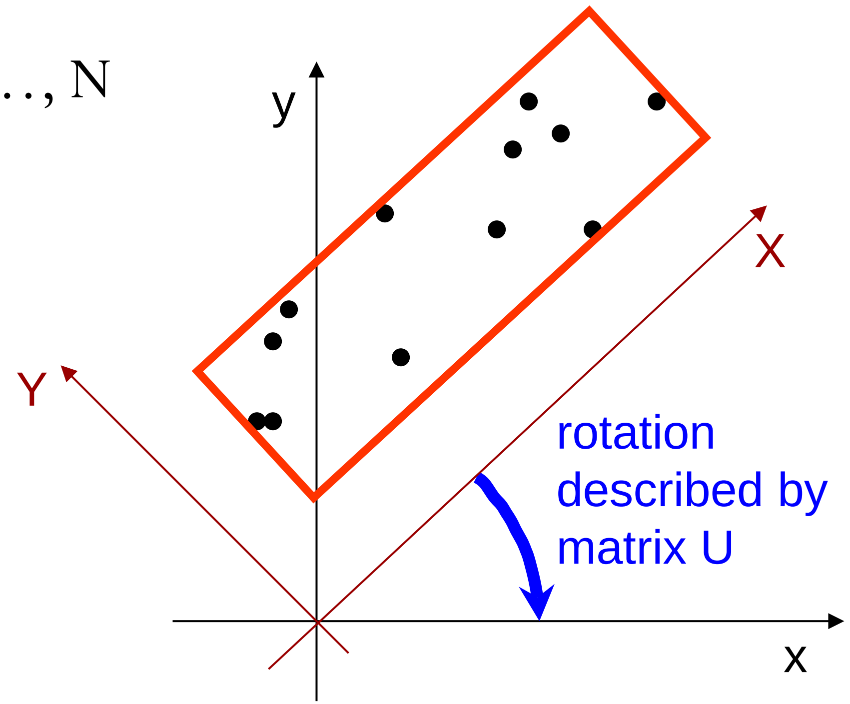


In 3D?



Compute OBBs

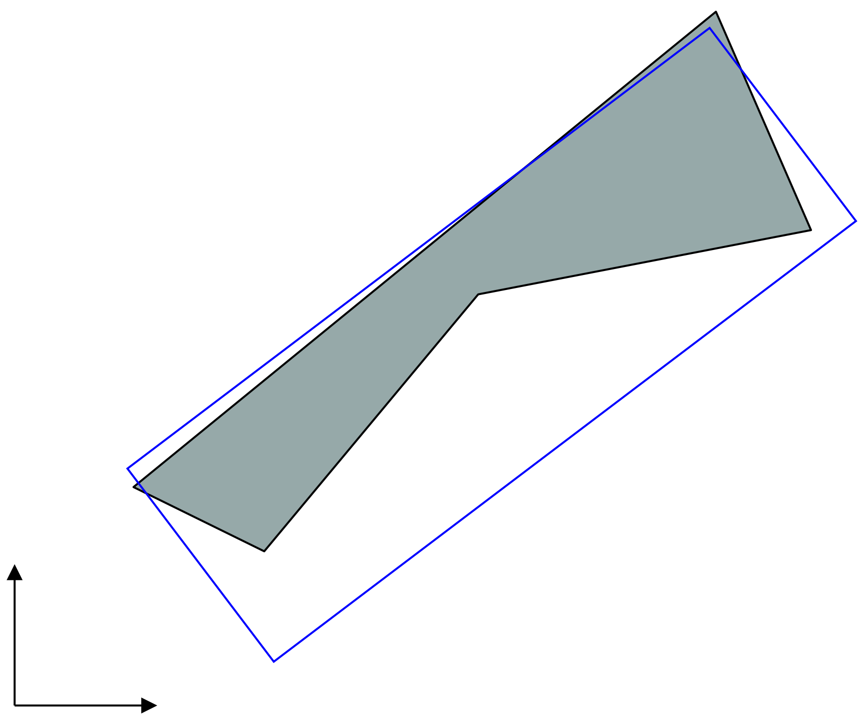
- N points $\mathbf{a}_i = (x_i, y_i, z_i)^T, i = 1,\ldots, N$
- SVD of $A = (\mathbf{a}_1\ \mathbf{a}_2 \ldots \mathbf{a}_N)$
  → $A = UDV^T$ where
  - $D = \mathrm{diag}(s_1,s_2,s_3)$ such that $s_1 \geq s_2 \geq s_3 \geq 0$
  - U is a 3x3 rotation matrix that defines the principal axes of variance of the $\mathbf{a}_i$'s
    → OBB's directions



- The OBB is defined by max and min coordinates of the $\mathbf{a}_i$'s along these directions
- Possible improvements: use vertices of convex hull of the $\mathbf{a}_i$'s or dense uniform sampling of convex hull

OBBs

- Invariant
- Less efficient to test
- Tight



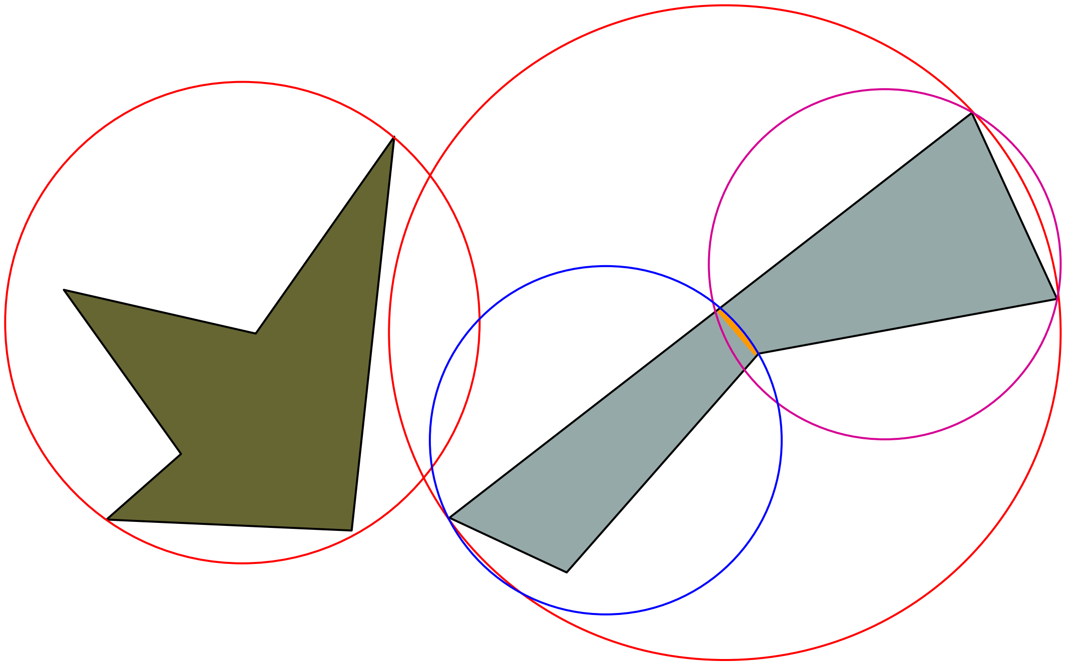
Comparison of BVs

	Sphere	AABB	OBB
Tightness	-	--	+
Testing	+	+	o
Invariance	yes	no	yes

No type of BV is optimal for all situations

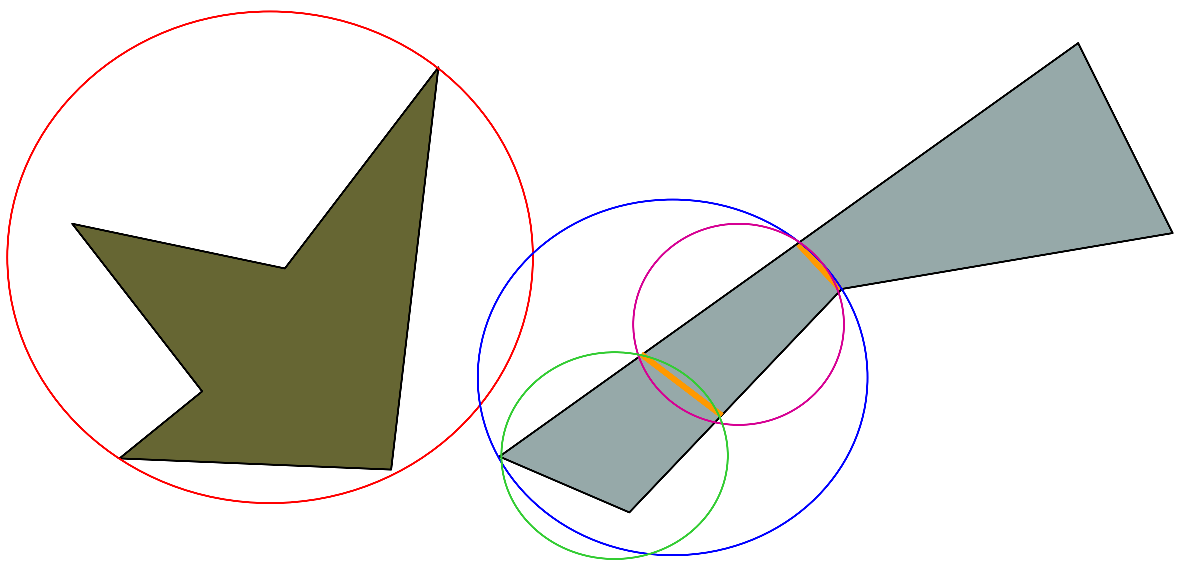
Bounding Volume Hierarchy (BVH)

- Bounding Volume Hierarchy method
  - Enclose objects into bounding volumes (spheres or boxes)
  - Check the bounding volumes first
  - Decompose an object into two



Bounding Volume Hierarchy (BVH)

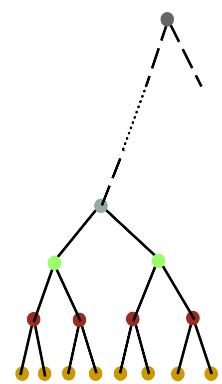
- Bounding Volume Hierarchy method
  - Enclose objects into bounding volumes (spheres or boxes)
  - Check the bounding volumes first
  - Decompose an object into two
  - **Proceed hierarchically**

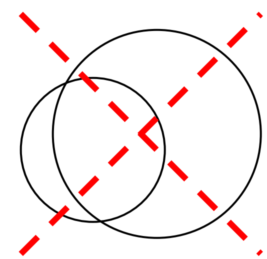


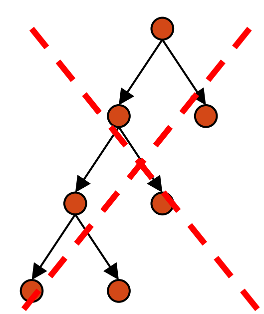
Bounding Volume Hierarchy (BVH)

- Construction
  - Not all levels of hierarchy need to have the same type of bounding volume
    - Highest level could be a sphere
    - Lowest level could be a triangle mesh

- Ideal BVH
  - Separation
  - Balanced tree

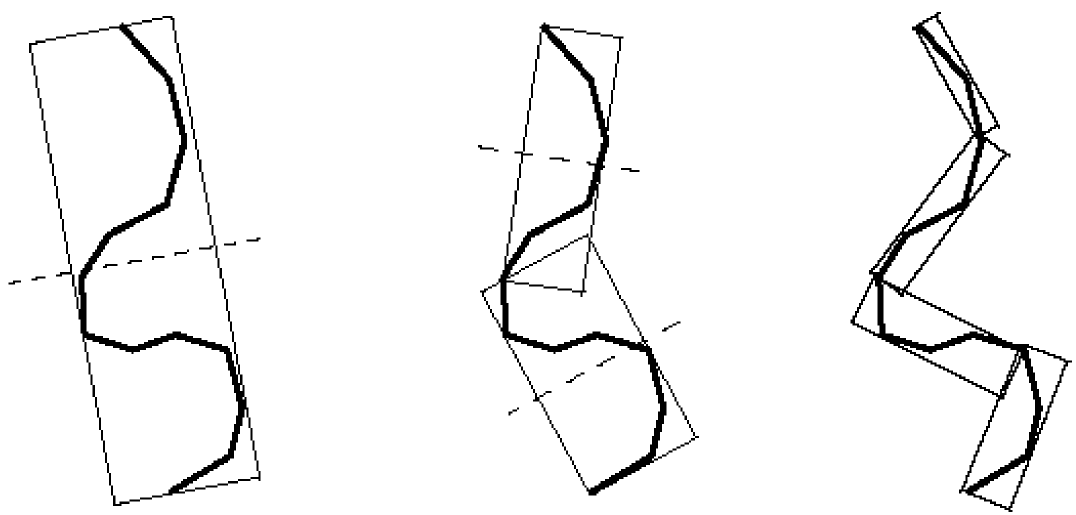




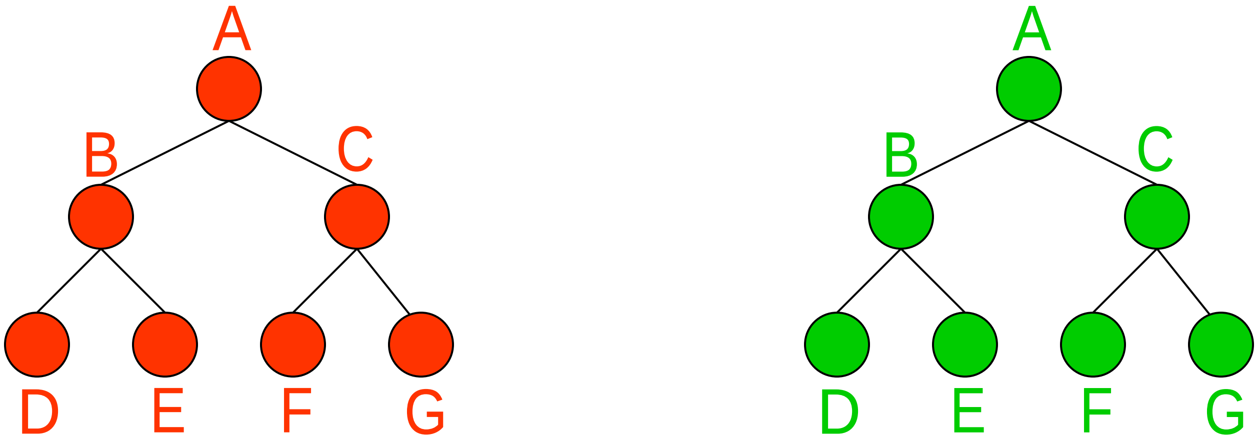


Construction of a BVH

- Strategy
  - Top-down construction
  - At each step, create the two children of a BV
- Example
  - For OBB, split longest side at midpoint



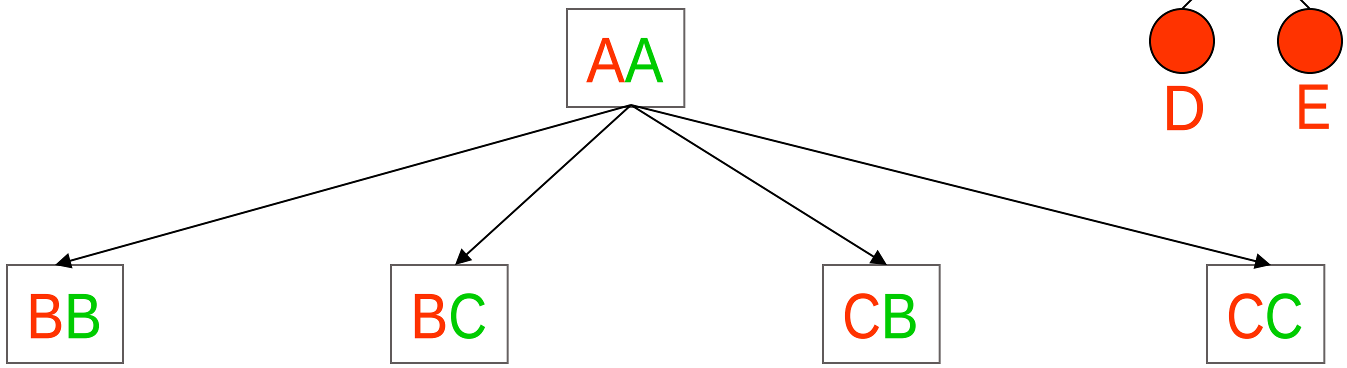
Collision Detection using BVH

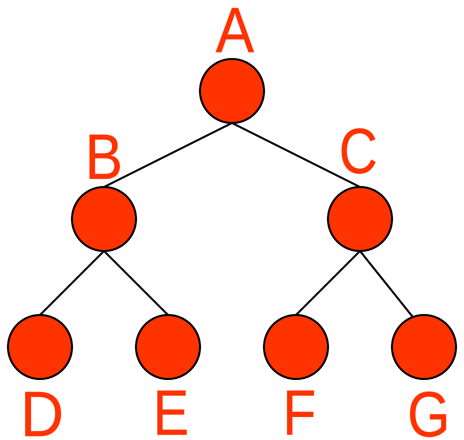


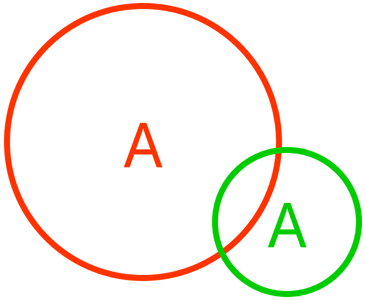
Two objects described by their precomputed BVHs

Collision Detection using BVH

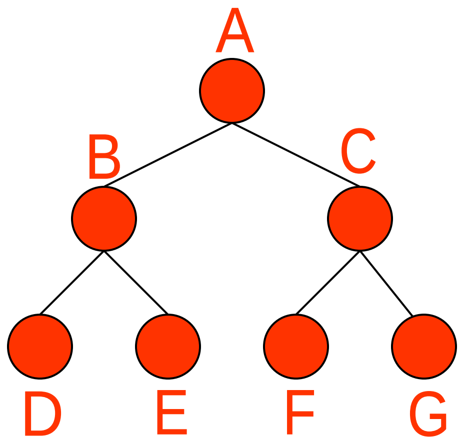
Search tree



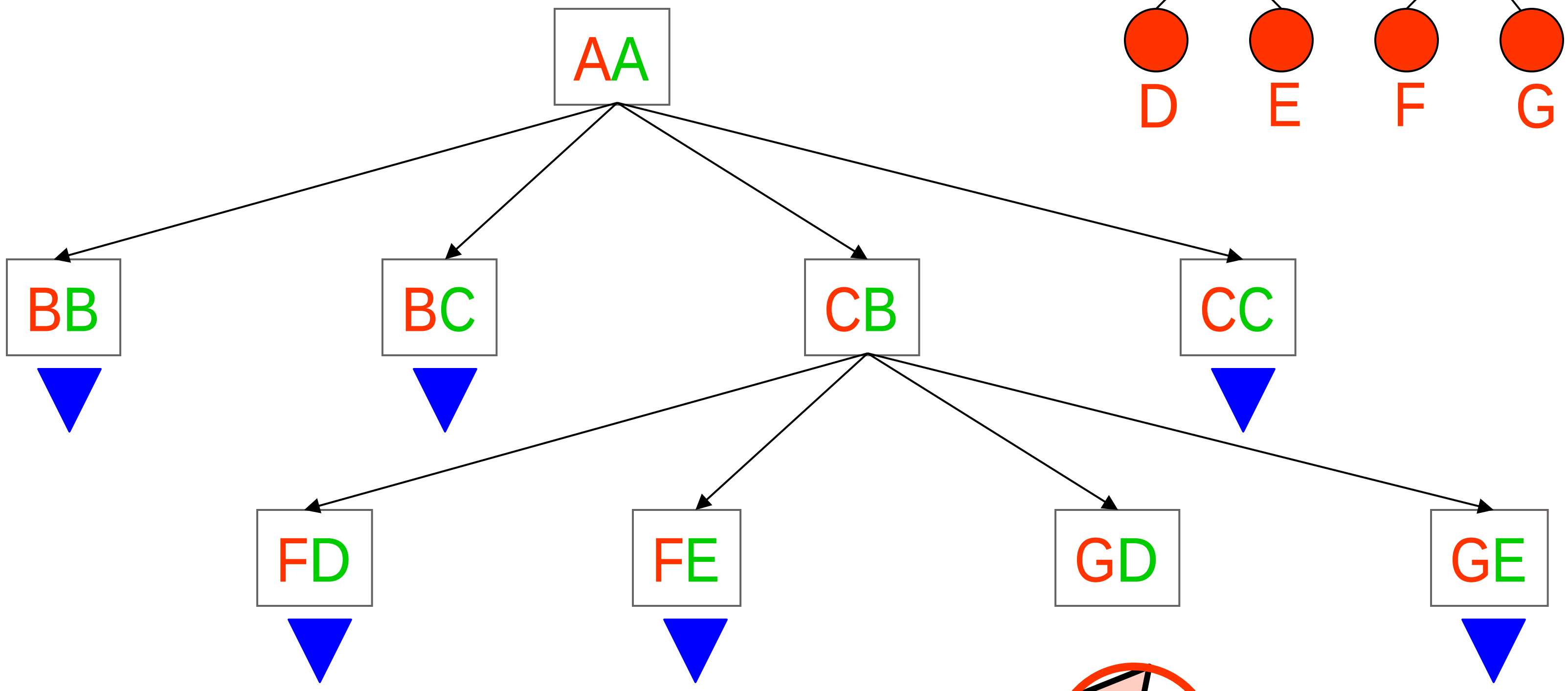


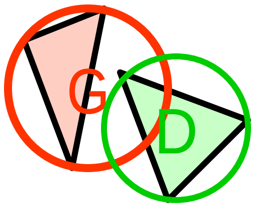


Collision Detection with BVH



Search tree



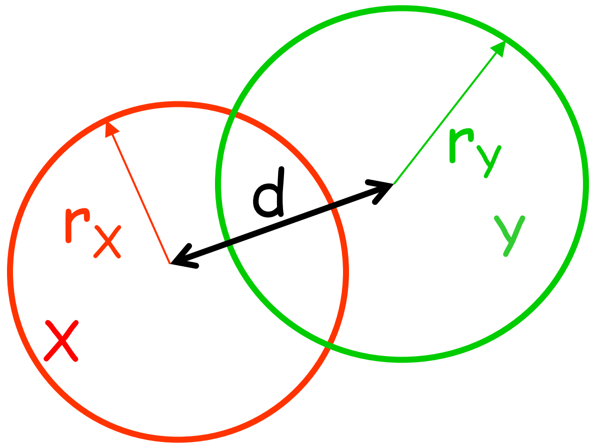


If two leaves of the BVH's overlap (here, **G** and **D**) check their content for collision

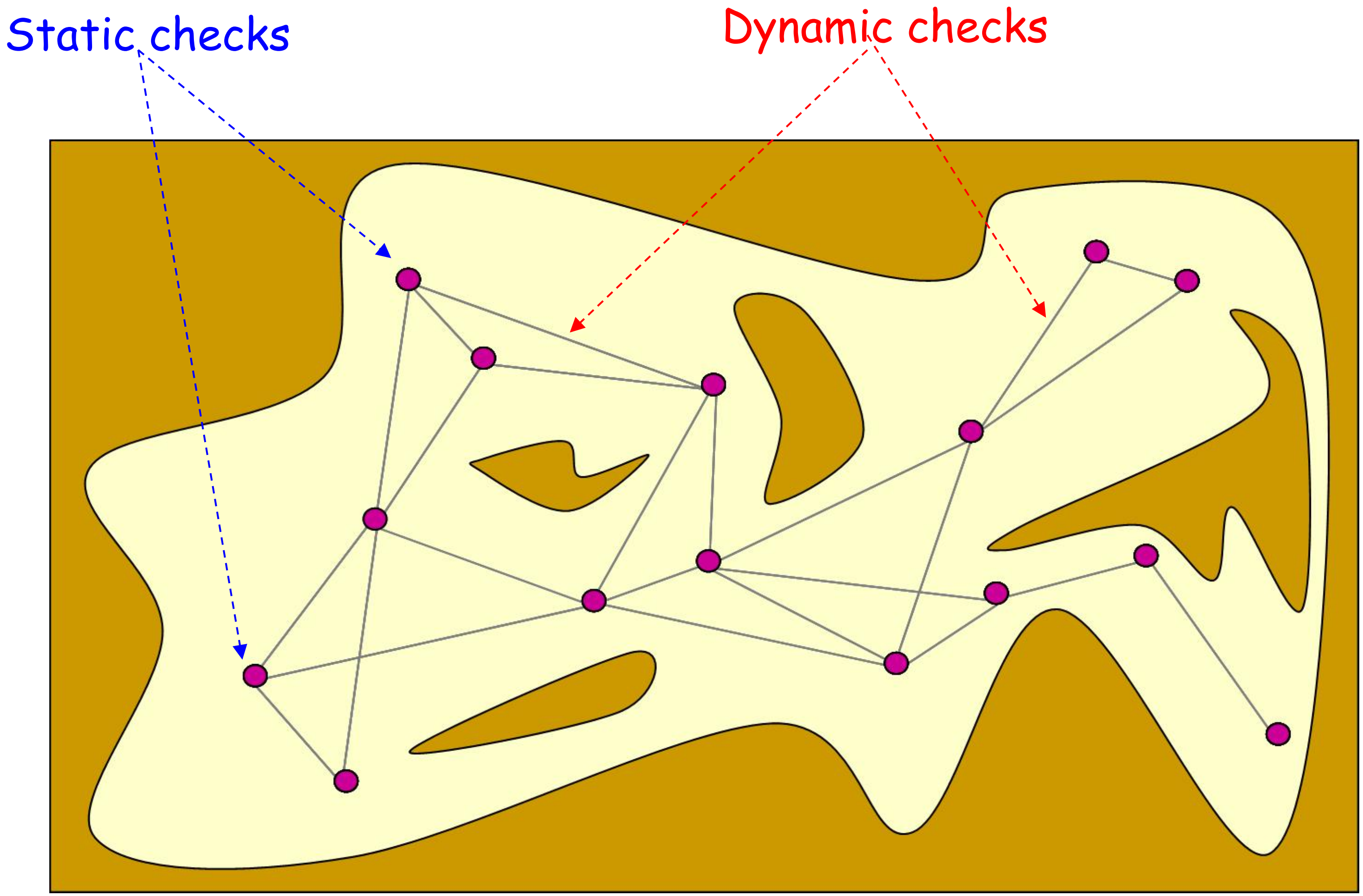
Search Strategy

- If there is collision
  - It is desirable to detect it as quickly as possible
- **Greedy best-first search strategy** with
  - Expand the node XY with largest relative overlap (most likely to contain a collision)
  - Many ways to compute distance **d**

$$\mathbf{f(N) = d/(r_X+r_Y)}$$

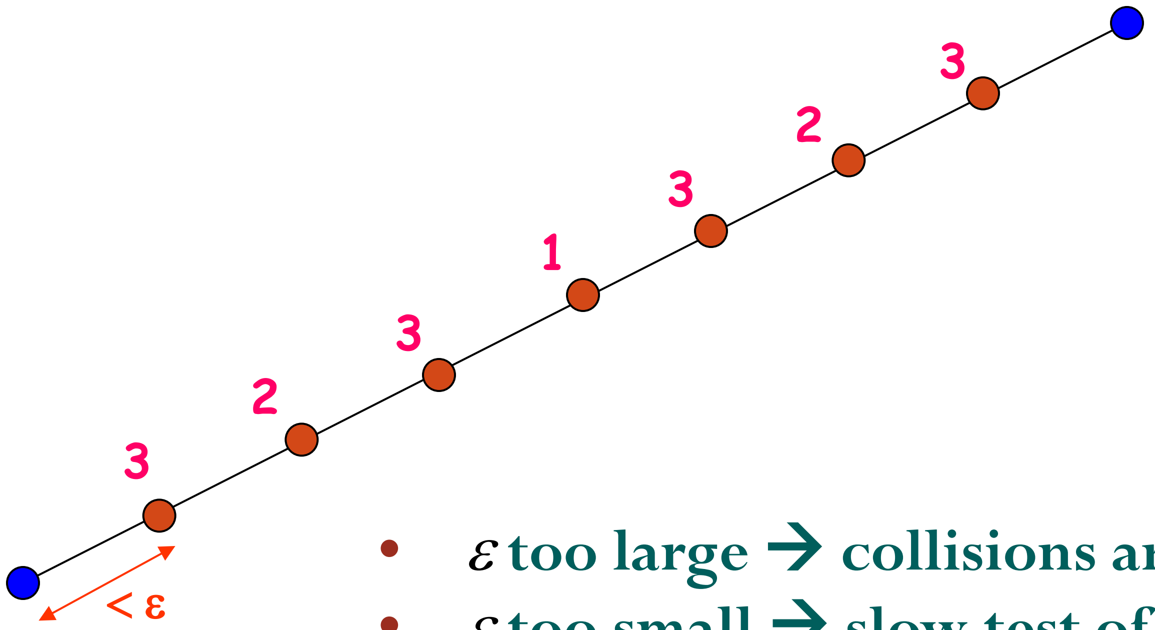


Static vs. Dynamic VS Collision Detection



Usual Approach to Dynamic Checking

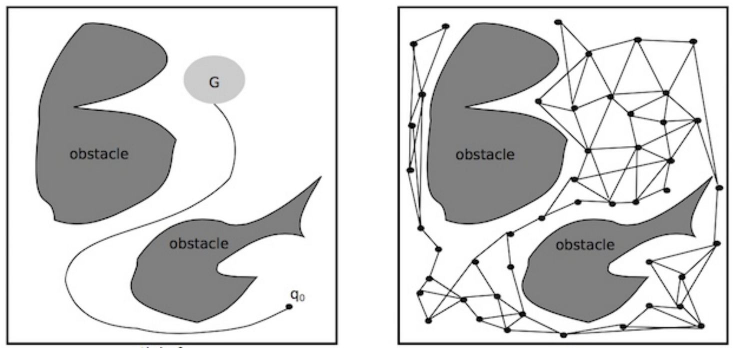
1) Discretize path at some fine resolution ε
2) Test statically each intermediate configuration



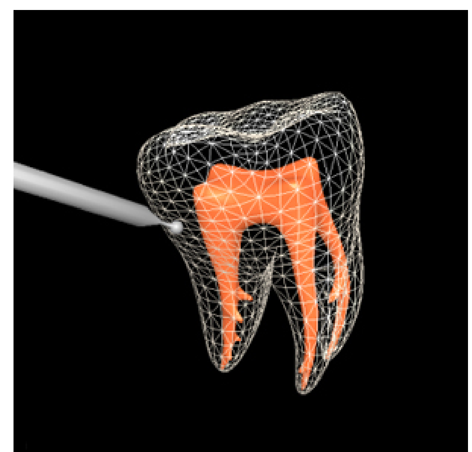
- **ε too large → collisions are missed**
- **ε too small → slow test of local paths**

Testing Path Segment vs. Finding First Collision

- PRM planning
  - Detect collision as quickly as possible → Bisection strategy



- Physical simulation, haptic interaction
  - Find first collision → Sequential strategy



Collision Checking for Moving Objects

- Feature Tracking
- Swept-volume intersection

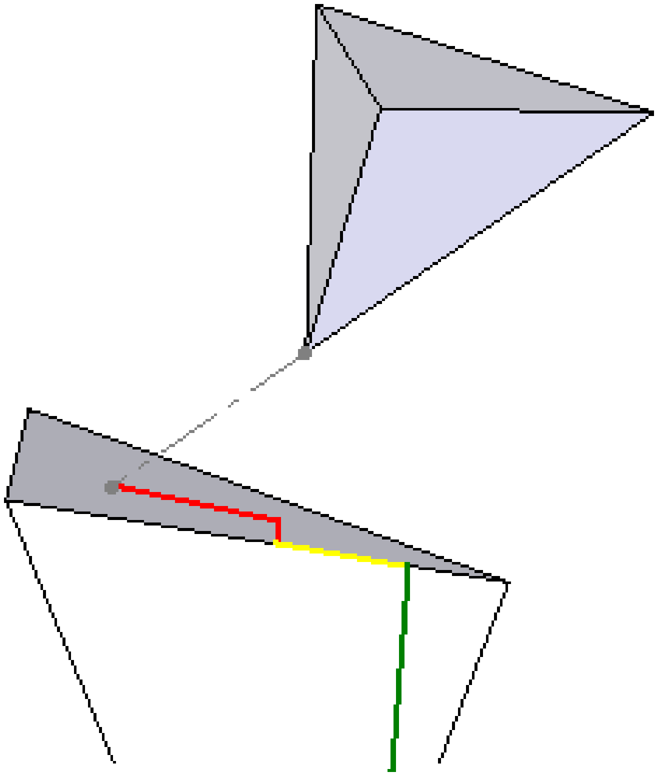
Feature Tracking

- Compute the Euclidian distance of two polyhedra
- Problem setup
  - Each object is represented as a **convex polyhedron** (or a set of polyhedra)
  - Each polyhedron has a field for its faces, edges, vertices, positions and orientations ← **features**
  - The closest pair of features between two polyhedra
    - The pair of features which contains the **closest points**
  - Given two polyhedra, find and keep update their **closest features** (see [1])

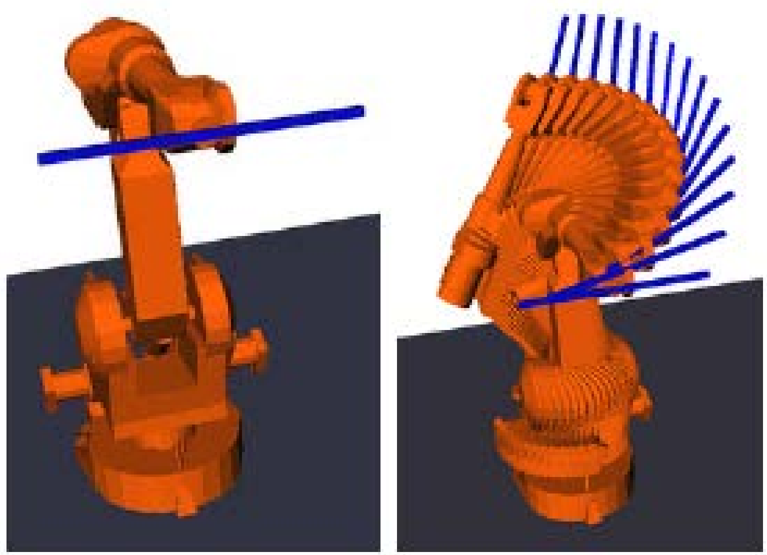
[1] M. Lin and J. Canny. A Fast Algorithm for Incremental Distance Calculation. Proc. IEEE Int. Conf. on Robotics and Automation, 1991

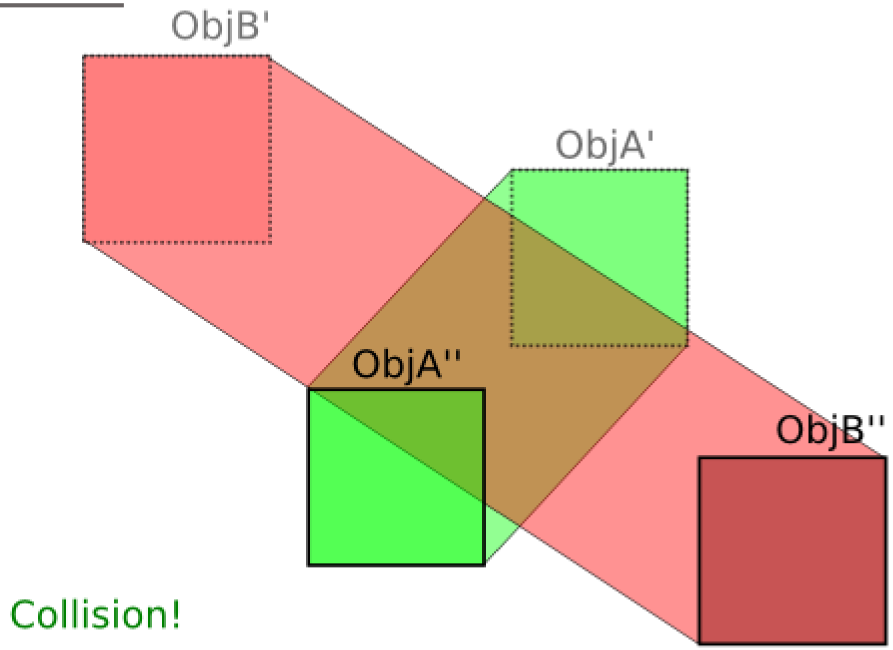
Feature Tracking

- Strategy
  - The **closest pair of features** (vertex, edge, face) between two polyhedral objects are computed **at the start configurations** of the objects
  - During motion, at each small increment of the motion, they are updated
- Efficiency derives from two observations
  - The pair of closest features changes relatively **infrequently**
  - When it changes the new closest features will usually be on a **boundary** of the previous closest features



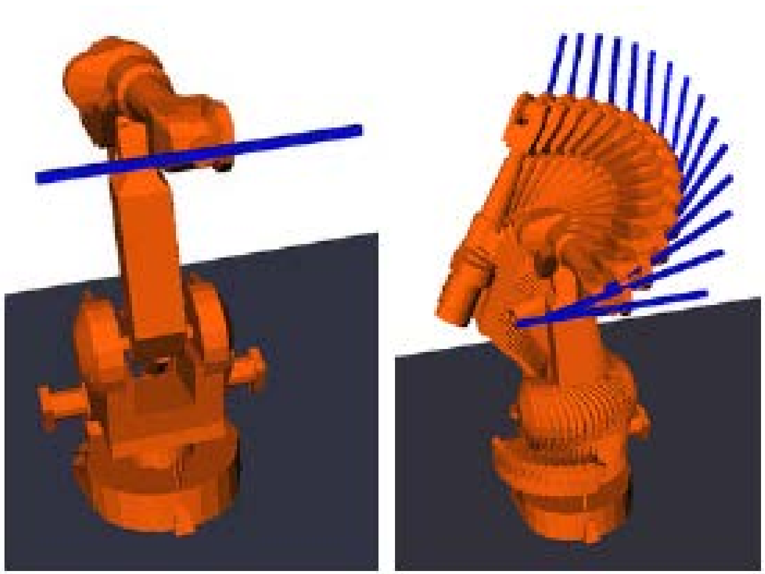
Swept-volume Intersection

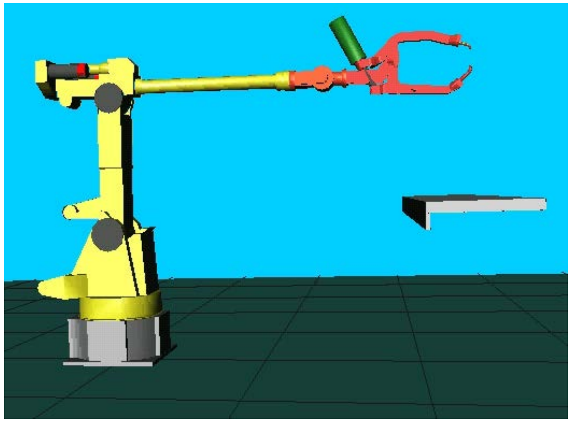


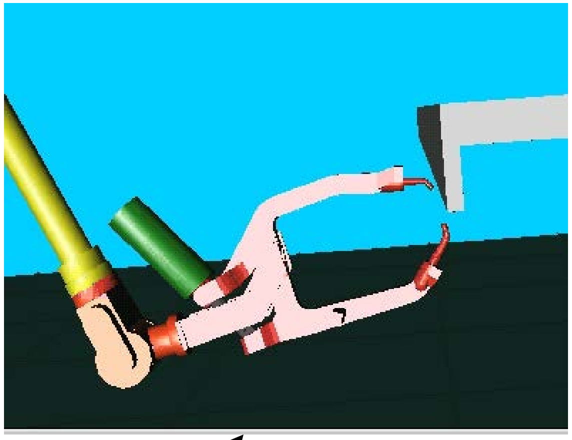


- ε too large → collisions are missed
- ε too small → slow test of local paths

Swept-volume Intersection







- ε too large → collisions are missed
- ε too small → slow test of local paths

Comparison

- Bounding-volume (BV) hierarchies
  - Discretization issue
- Feature-tracking methods
  - Geometric complexity issue with highly non-convex objects
- Swept-volume intersection
  - Swept-volumes are expensive to compute. Too much data.

Readings

- Principles CH7
- Review of Probability Theory
  - https://drive.google.com/file/d/0B7SwE0PHMbzbU1FqcnNORTFZOWM