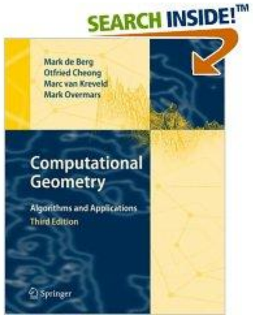
Collision Detection

Simon Leonard

Hands On Resources



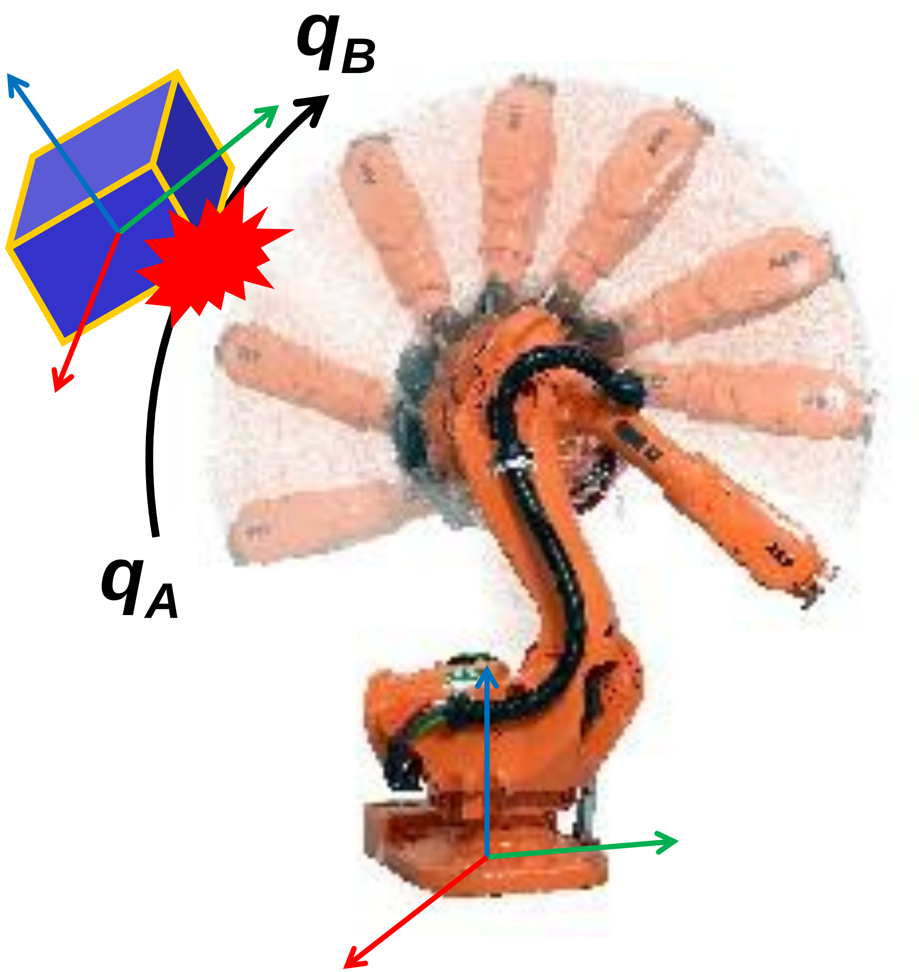
Computational Geometry
Computational Geometry: Algorithms and Applications

Collision Detection
Real-Time Collision Detection

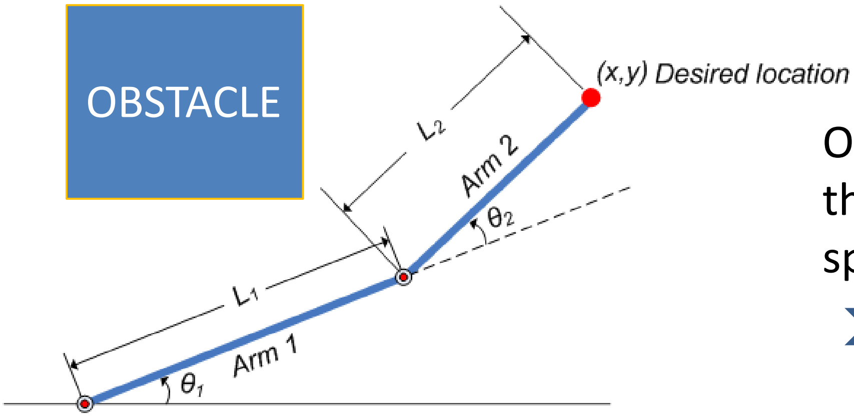
Flexible Collision Library (FCL)
http://gamma.cs.unc.edu/FCL

Motion Planning

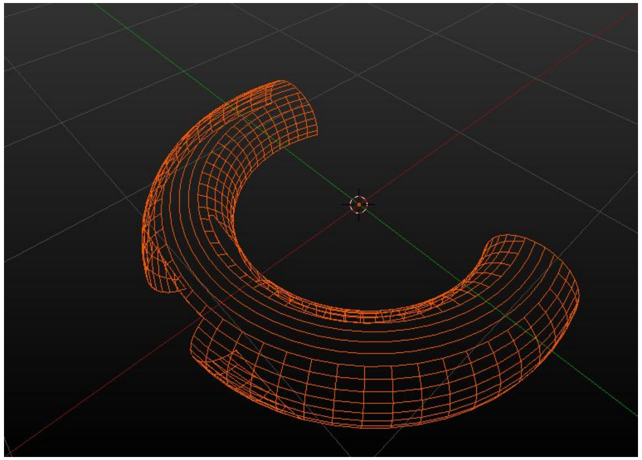
- Robot moves in configuration space
- Objects and distances are defined in Cartesian space
- Motion planning searches for collision free paths
- If a robot moves from q_A to q_B, how do you determine if it will hit anything along the way?
  - What is “the way”?
  - Reactive vs interpolated

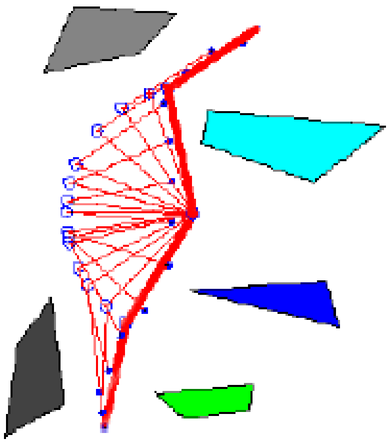


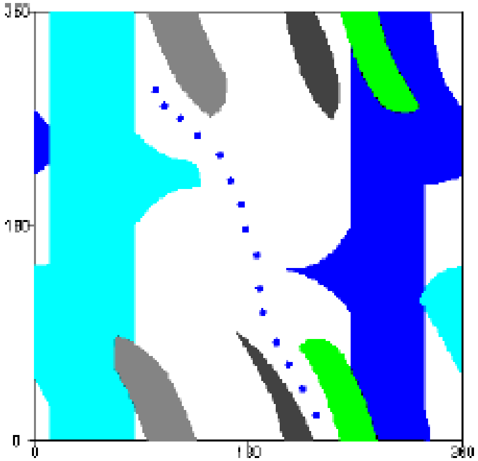
Motion: Configuration Space
Obstacles: Cartesian Space



Obstacle carving the configuration space







Principles of Robot Motion

Motion: Configuration Space
Obstacles: Cartesian Space

- Reactive system
  - Use sensors to avoid or mitigate contacts
    - Range finder to detect distances to obstacles
    - Bumpers/tactile sensors to detect small collisions
    - Force sensors to detect unexpected interactions
  - Use a “greedy” algorithm (like a potential field) to move toward a goal while avoiding the obstacle
- Planning system
  - Use the 3D geometry of the robot and environment to plan a path before moving
  - Find where the robot can move without colliding

Reactive Motion

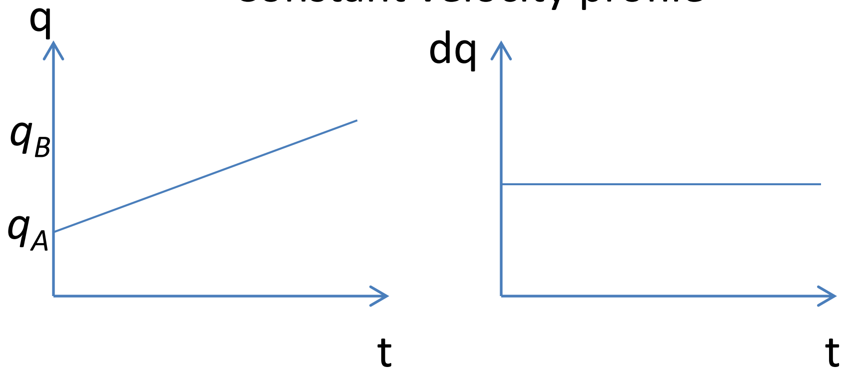
- Control and sensor-based
- A robot can move in reaction to an observation
  - Typically obtained from measurements (encoders, GPS, camera, range, force, bumper, etc.)
- Given a measurement of how far the robot is from the goal, the robot takes a step toward the goal

www.dlr.de

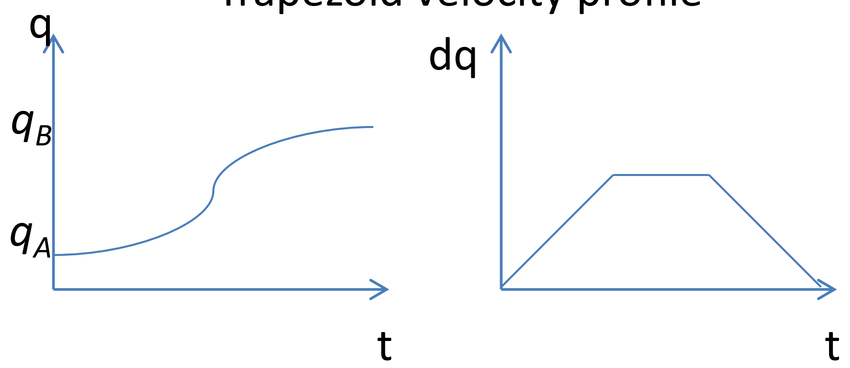
Interpolated Motion
How Robots Move?

- Parametric motion (i.e. a joint follows a polynomial trajectory)
- A robot moves according to a known equation that specifies at time varying configuration, velocity and acceleration in configuration space
  - Linear, trapezoid, quintic, etc.

Constant velocity profile

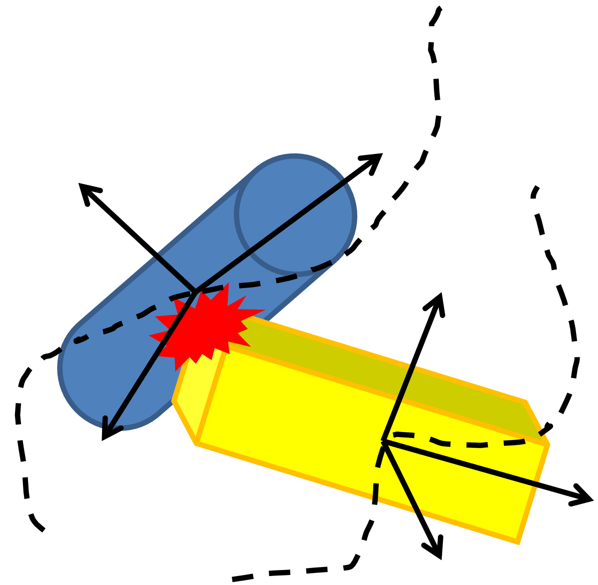


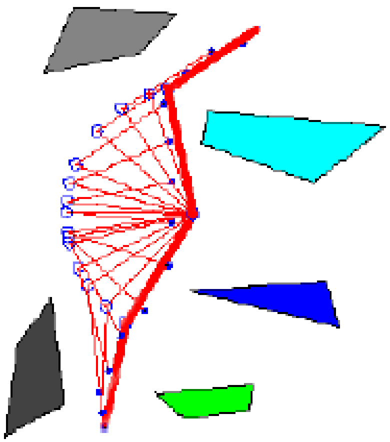
Trapezoid velocity profile

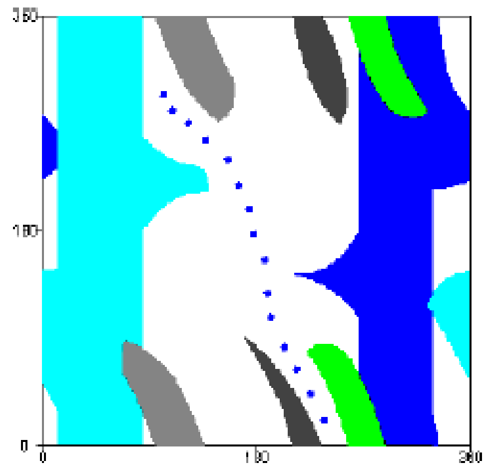


Interpolated Motion

- If we know that a robot moves from q_A to q_B according to a parametric trajectory, how do we determine if (and where) the robot collides with an obstacle?
- Larger question: if an obstacle travels in space, how do we determine if (and where) it will collide?
- What if the robot and obstacle both move?



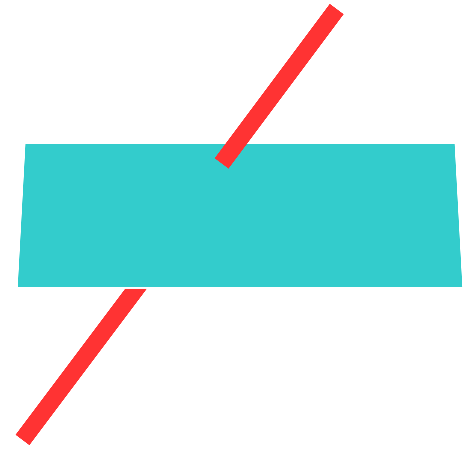


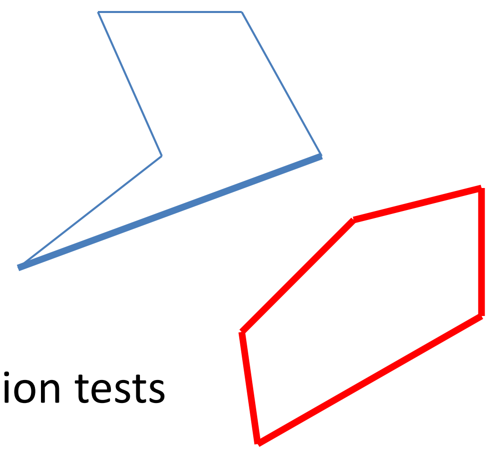


Principles of Robot Motion

Geometry: 3D Meshes

- Collision detection between basic shapes (circles, spheres, lines, triangles and squares and 3 boxes) is fairly easy
- Complex shapes can be composed of several thousands of simple shapes and doing an exhaustive search is costly

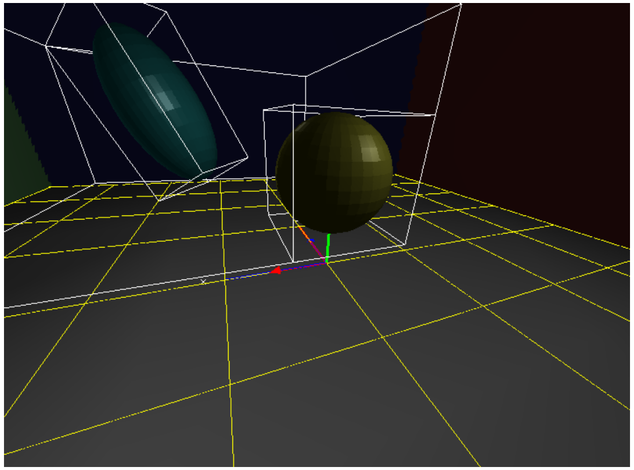




25 collision tests

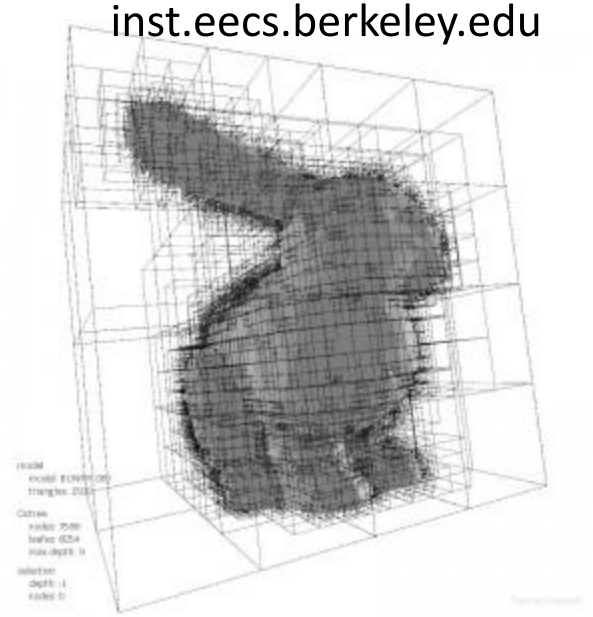
Collision Detection

- Using basic shapes to "bound" objects is conservative



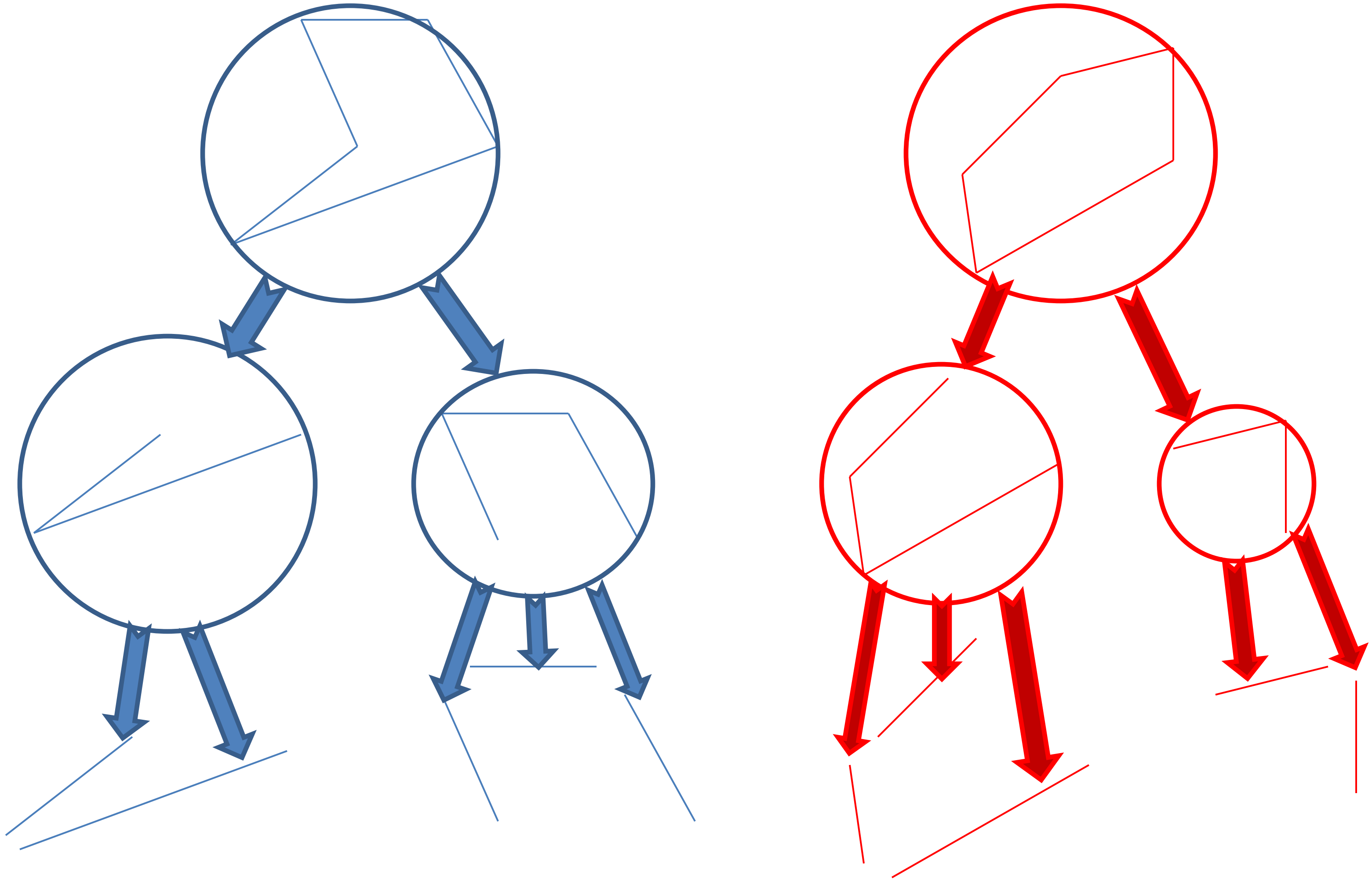
inst.eecs.berkeley.edu

- Organize basic shapes in a hierarchy of bounding volumes



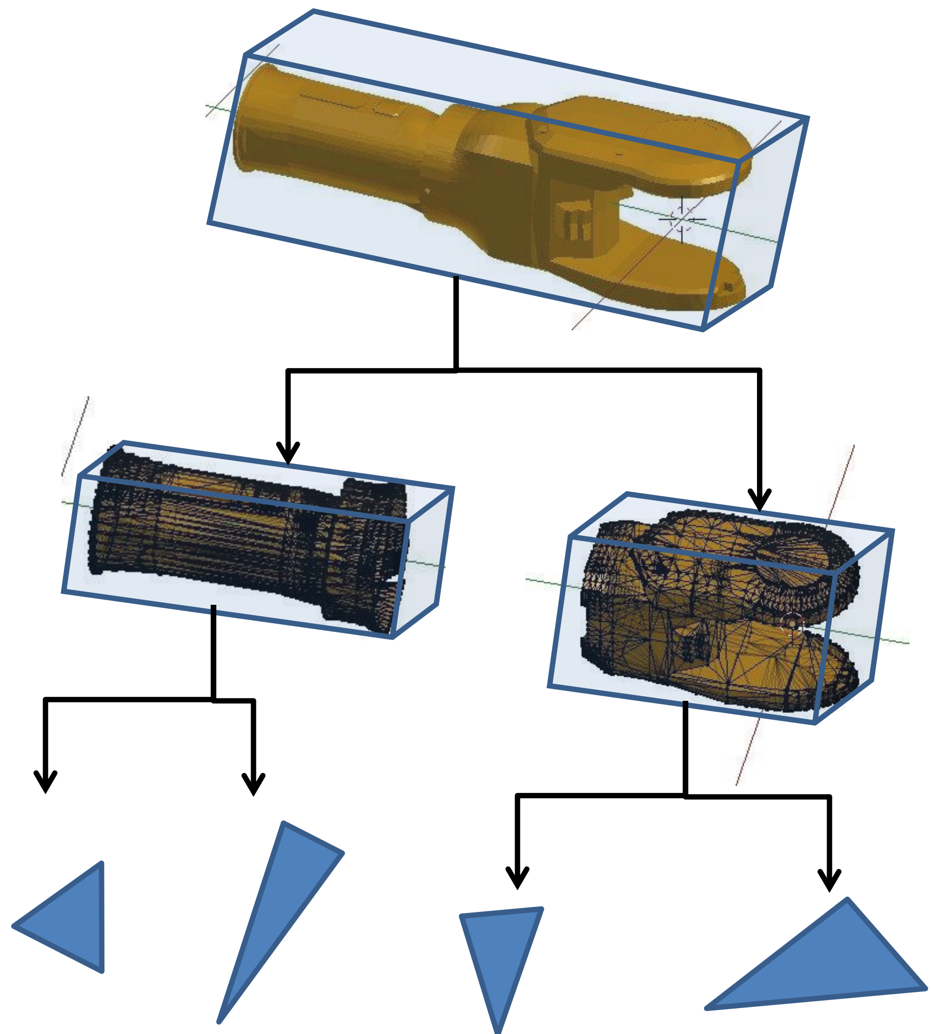
thomasdiewald.com

Hierarchy of Bounding Volumes



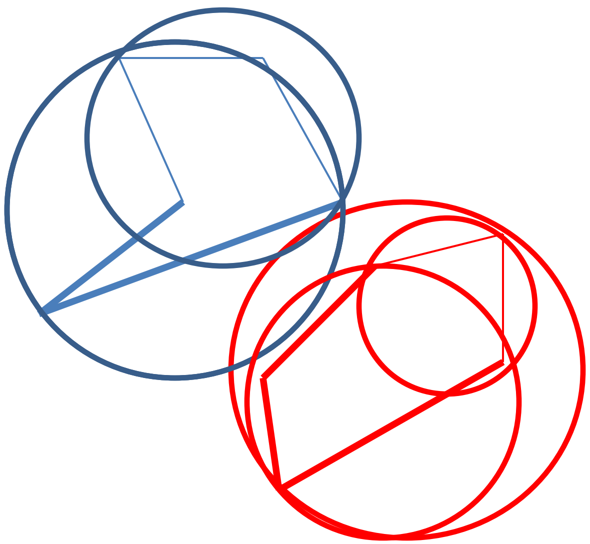
Bounding Volumes Hierarchy

- More complex models require hierarchies of bounding volumes
  - Spheres
  - Axis Aligned Bounding Boxes (AABB)
  - Oriented Bounding Boxes (OBB)
  - Swept Sphere Volumes (SSV)
- Unless the geometry changes, build the hierarchy once (offline)
- What makes a good bounding volume?
  - Tightness of fit
  - Speed of collision detection computation between bounding volumes



Collision Detection

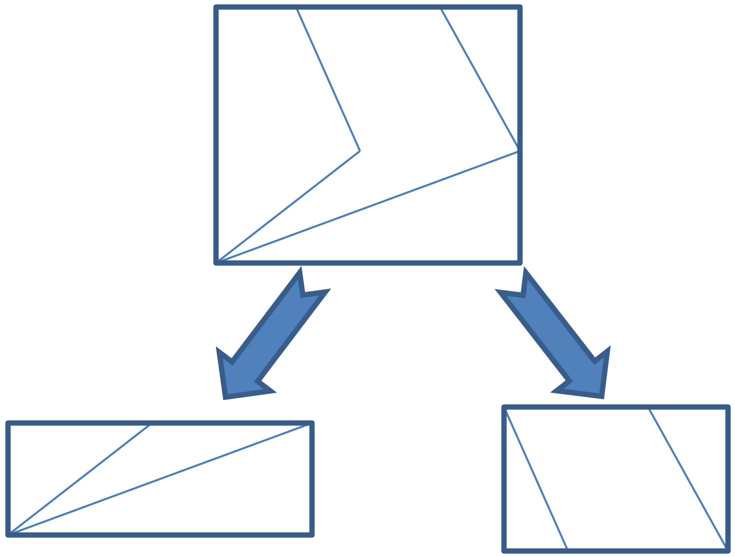
- Given the hierarchies of two objects
  - Check if the top level bounding volumes collide
    - If they don't collide then the objects do not collide
    - If they collide then test for collision between the children
  - Apply recursion until we a collision is found between two primitives (triangles) or no more collision test are needed

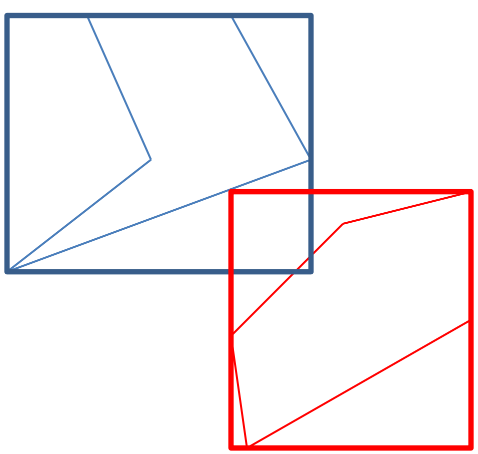


11 collision tests

Axis Aligned Bounding Box (AABB)

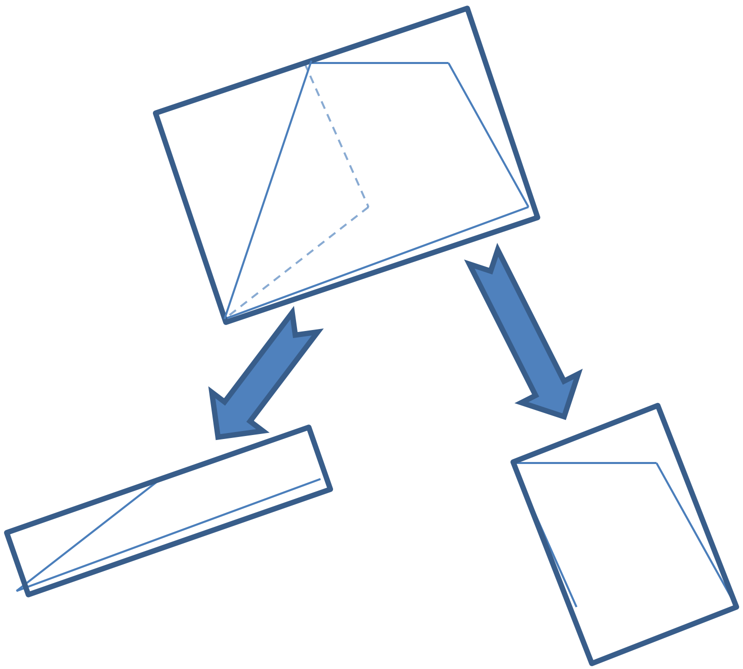
- Bound the volume with a 3D box that is aligned with the X-Y-Z axis
  - Easy to build
  - Not very tight fit
  - Fast to test for collision





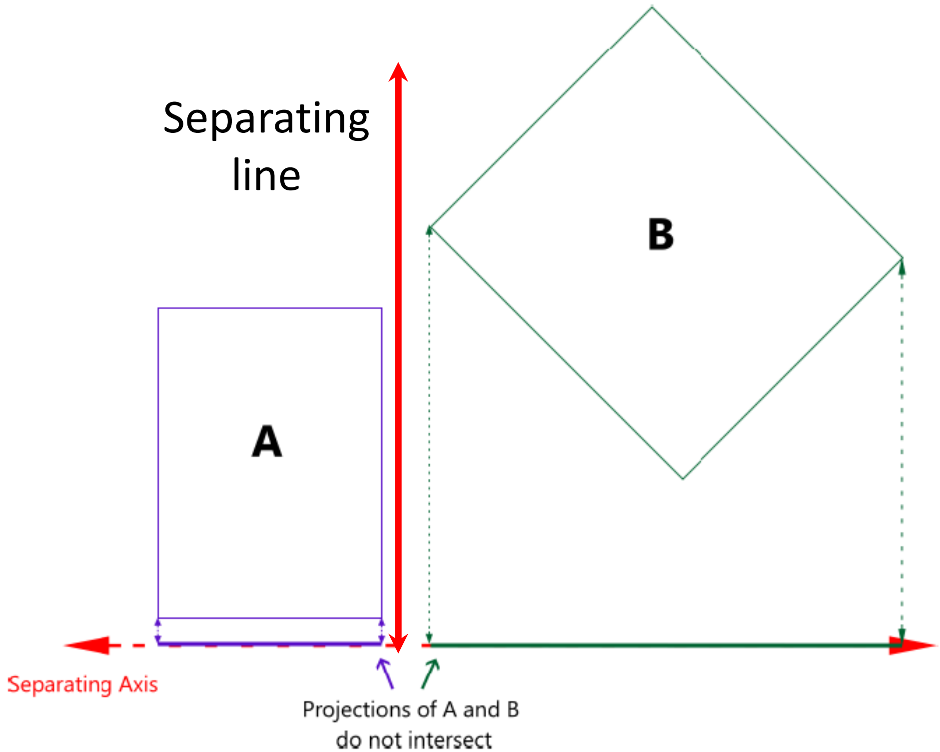
Oriented Bounding Box (OBB)

- Keep the vertices of the mesh’s convex hull
- Find the principal axis of the vertices
  - This gives an orientation of the bounding volume
- Divide the mesh along the dominant axis



Collision Between OBB

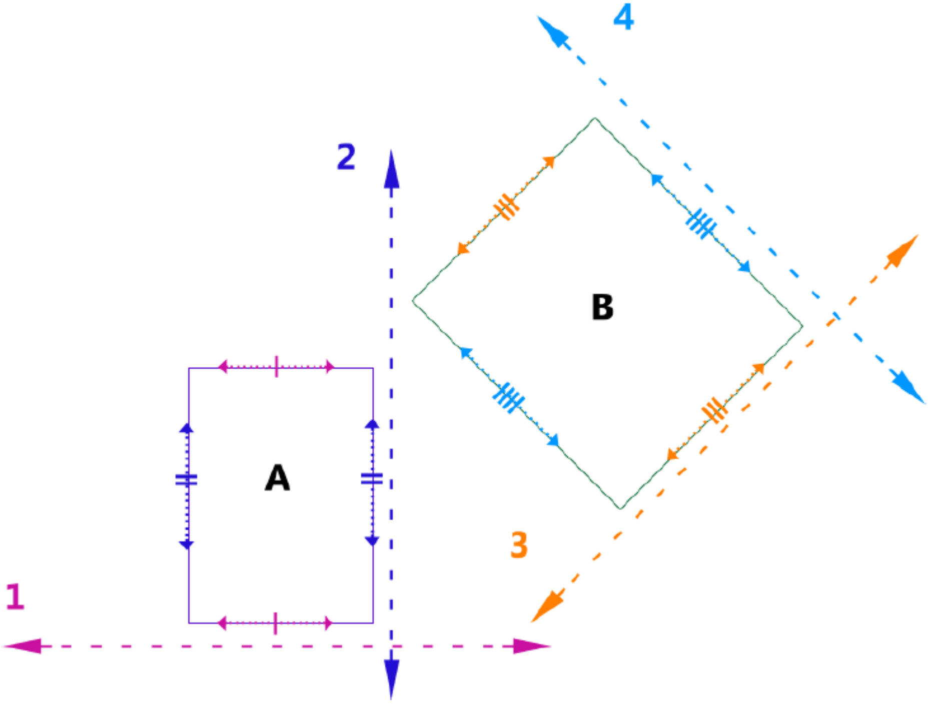
- Separating Axis Theorem
  - Two OBB do not collide if there is a separating *axis* L on which the projection of both OBB does not intersect
- How do we find this line?
  - Note that the separating *line* is perpendicular to the separating axis
  - A separating line exists if and only if there is a separating line that is parallel to an edge of rectangle A or B



http://www.jkh.me

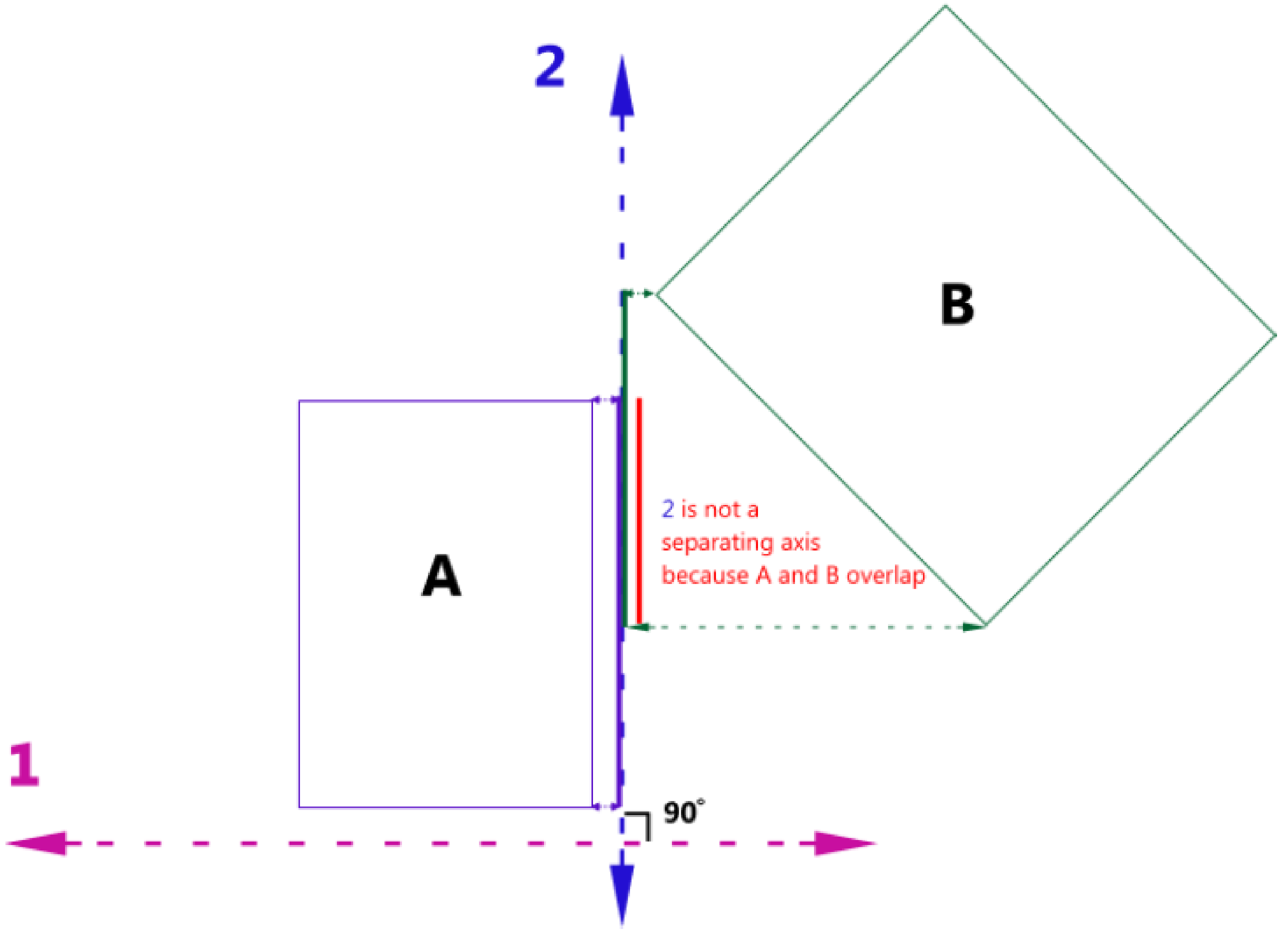
Collision Between OBB

- Use separating lines that are parallel to the edges of A and B
- Given that each rectangle has 2 parallel edges only 4 axis are checked
- Project both rectangles on each axis and check if the projections intersect



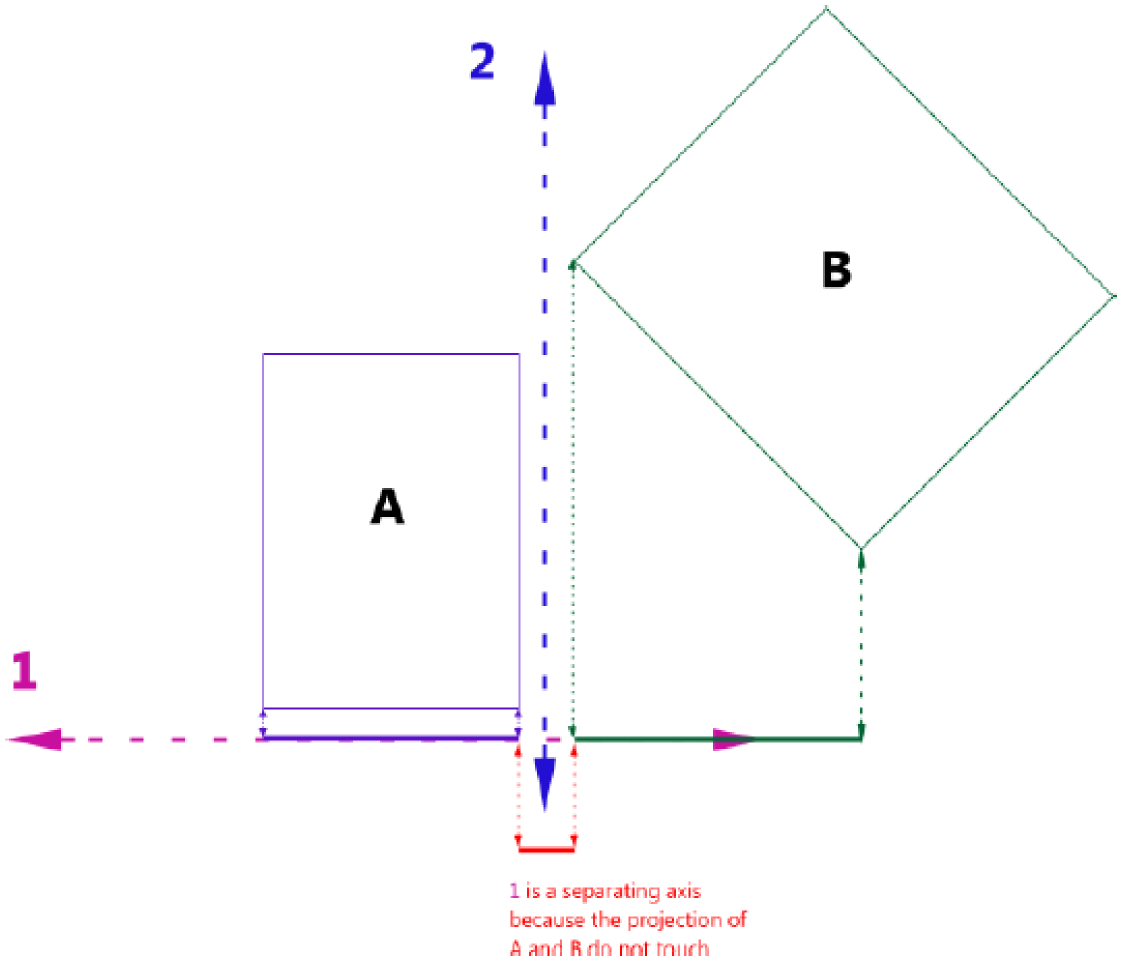
http://www.jkh.me

Collision Between OBB



http://www.jkh.me

Collision Between OBB

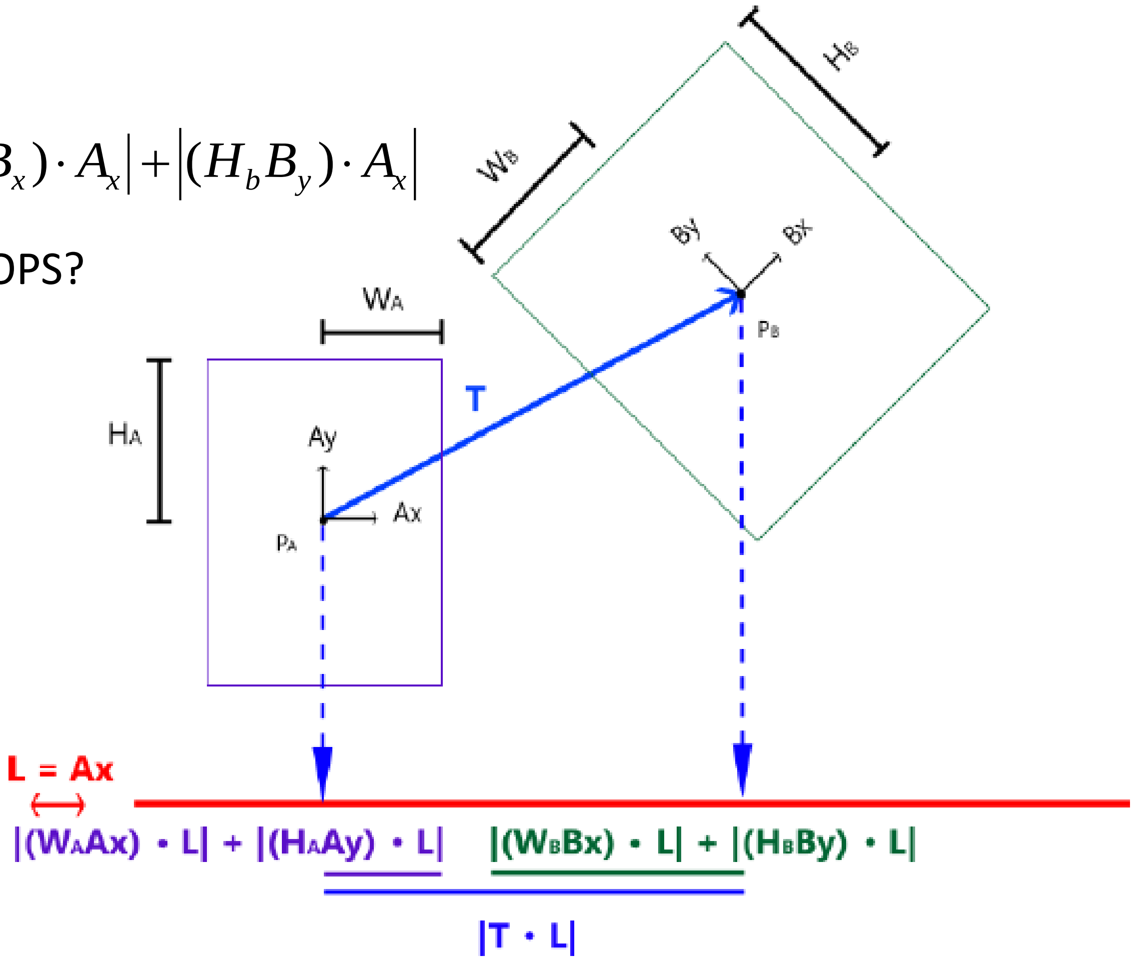


http://www.jkh.me

Collision Between OBB

$$\left|T \cdot A_x\right| > W_a + \left|(W_B B_x) \cdot A_x\right| + \left|(H_b B_y) \cdot A_x\right|$$

How many FLOPS?



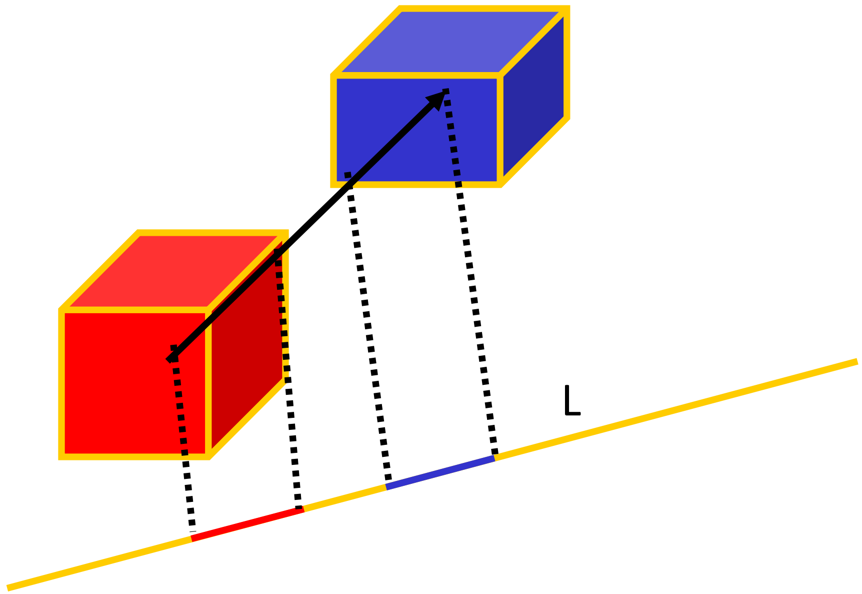
http://www.jkh.me

Collision Between OBB

- Separating Axis Theorem
  - Two OBB do not collide if there is a separating line L on which the projection of both OBB does not intersect.
  - Test for 15 axes is sufficient to determine if such line exists:

Collision between faces
    - 3 axes of A
    - 3 axes of B

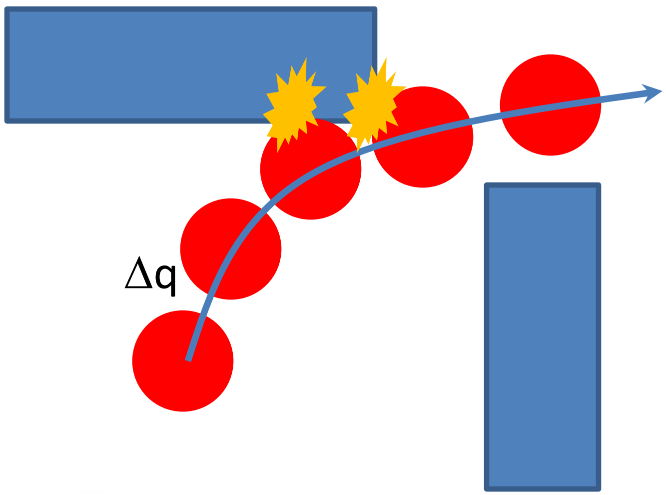
Collision between edges
    - $x_a \times x_B$, $x_a \times y_B$, $x_a \times z_B$
    - $y_a \times x_B$, $y_a \times y_B$, $y_a \times z_B$
    - $z_a \times x_B$, $z_a \times y_B$, $z_a \times z_B$



- 200 FLOPS max!

Using Collision Detection During a Trajectory

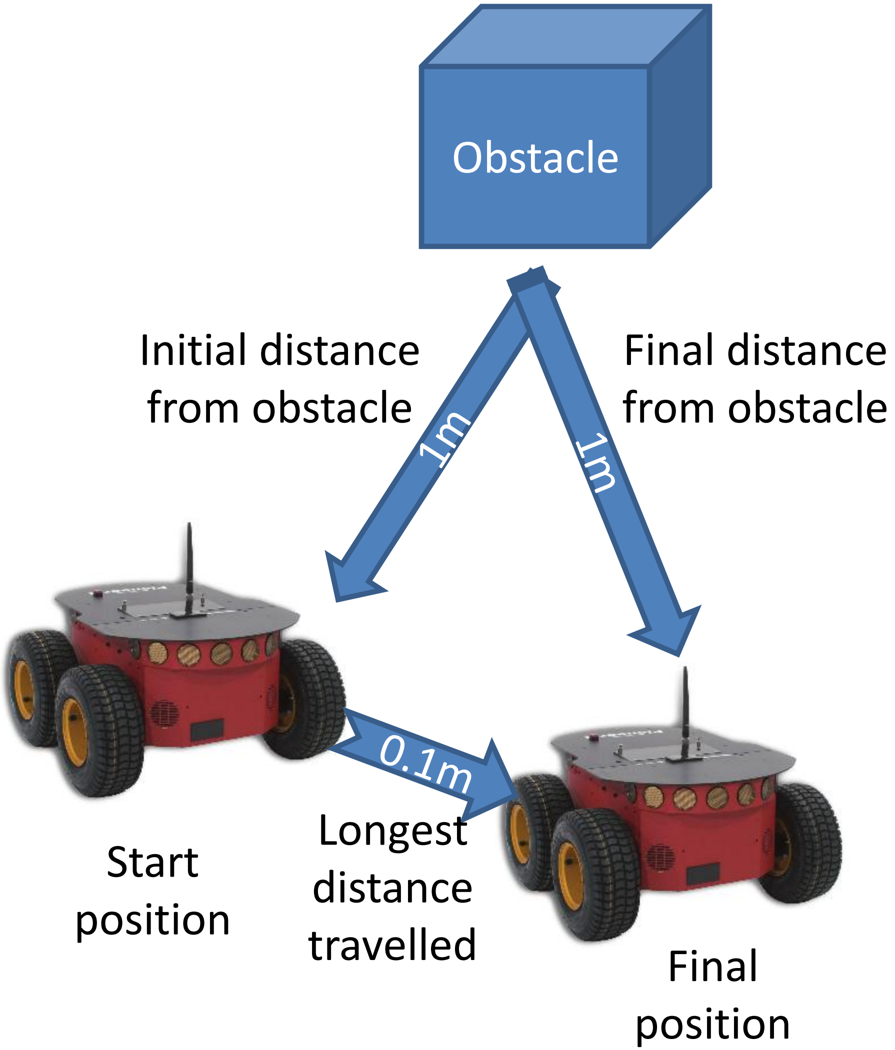
- If we know that a robot moves from q_A to q_B according to an parametric trajectory, how do we determine if (and where) the robot collides with an obstacle?
  1. Move the robot from q_A to $q_A+\Delta q$ and test for a collision between te robot and its environment
  2. Repeat until the robot reaches q_B
- How large should Δq be?
  - If Δq is too large we might step over thin objects
  - If Δq is tool small more tests will be used



Schwarzer 2005

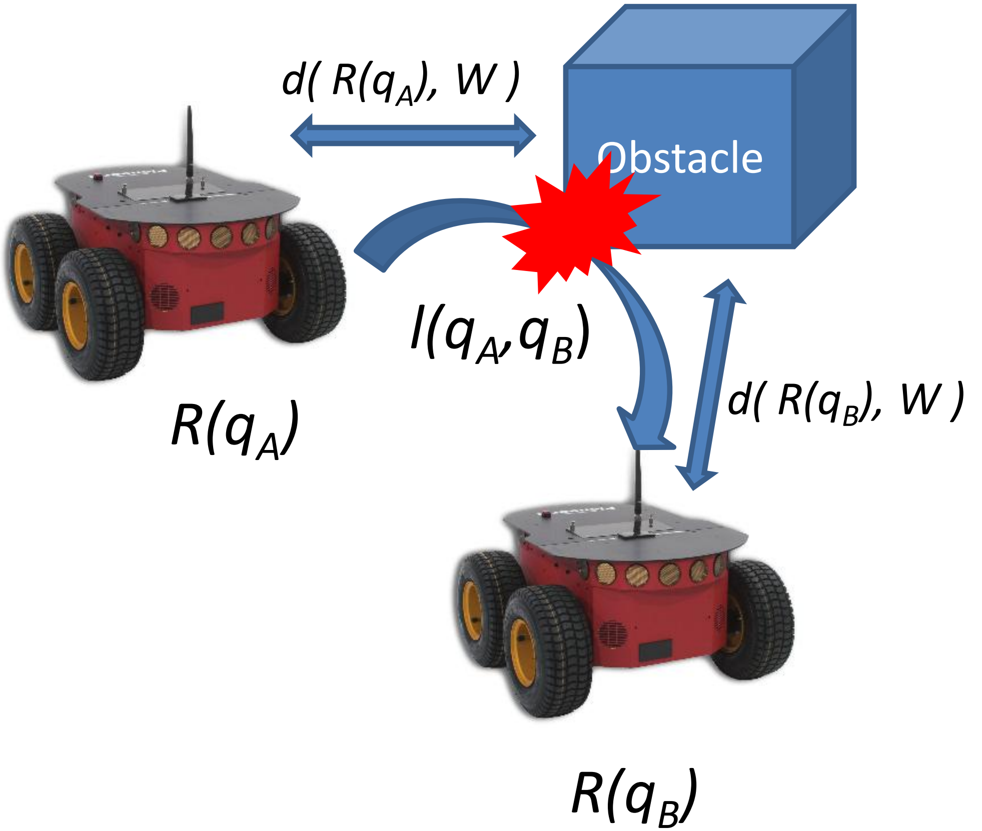
Adaptive Local Planner

- What is the relation between the initial and final distances to collision and the maximum travelling distance?
  - I start 1m away from any obstacle
  - I finish 1m away from any obstacle
  - No point on me (robot) travelled by a distance greater than 0.1m
  - Can I determine that I did not collide?



Adaptive Local Planner

- Suppose there is a collision between the robot R and the world W when the robot moves from q_A to q_B and that the collision happens at configuration q_C
- Then let
  - *d(R(q), W)*: The shortest distance between the robot in configuration *q* and any obstacle in *W*.
  - *l(R(qA), R(qB))*: The longest distance travelled by any point on the robot as it moves from q_A to q_B



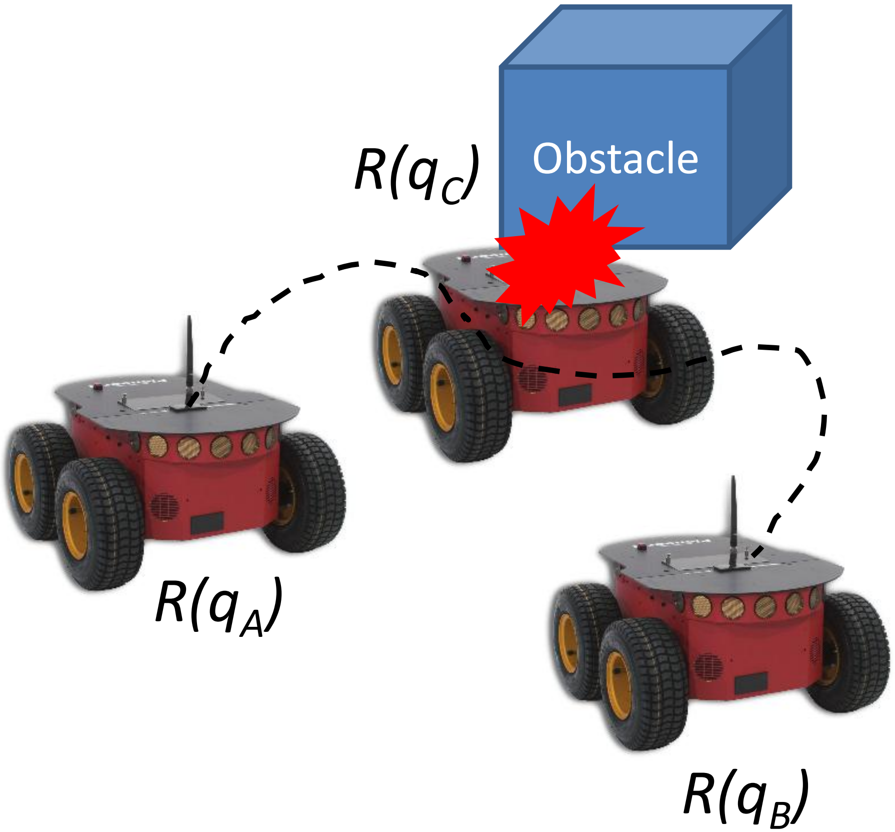
Adaptive Local Planner

- Suppose there is a collision between the robot and the world at configuration q_C
- Then it must be that

$$d(\, R(q_A), W \,) < l(\, R(q_A), R(q_C) \,) \quad (1)$$

$$d(\, R(q_B), W \,) < l(\, R(q_B), R(q_C) \,) \quad (2)$$

1) From q_A to q_C, there is a point that travels a greater distance than the shortest *initial* distance between the robot and the obstacle
2) From q_B to q_C, there is a point that travels a greater distance than the shortest *final* distance between the robot and the obstacle



Adaptive Local Planner

- If we add (1) and (2) we can determine that there is no collision between q_A and q_B if

$$d(\, R(q_A), W \,) + d(\, R(q_B), W \,) > l(\, R(q_A), R(q_B) \,)$$

- No need to find the collision configuration q_C!
- If the inequality is not satisfied?
  - It does not mean that there is a collision
  - Divide the trajectory $[q_A, q_B]$ in two $[q_A, q_M]$ and $[q_M, q_B]$ and test each of them recursively.
  - Only need to test for a collision at q_A, q_B and q_M

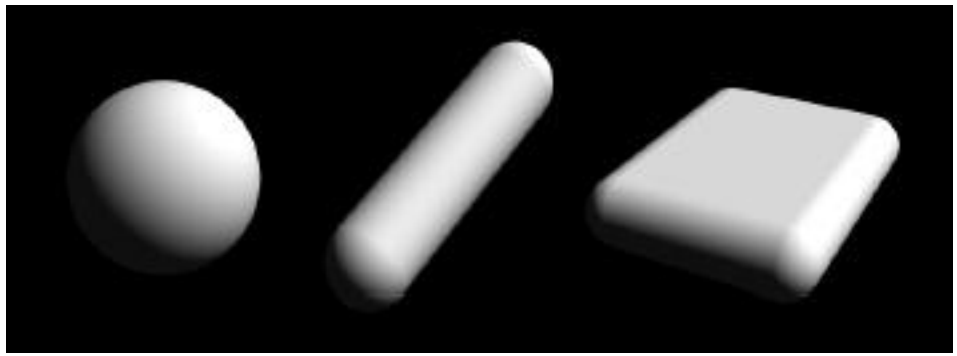
Adaptive Local Planner

- If the robot is far from any obstacle and does a small motion, then $d(R(q_A),W)+d(R(q_B),W)$ is large and $l(R(q_A), R(q_B))$ is small
- Therefore

$$d(R(q_A), W) + d(R(q_B), W) > l(R(q_A),R(q_B))$$

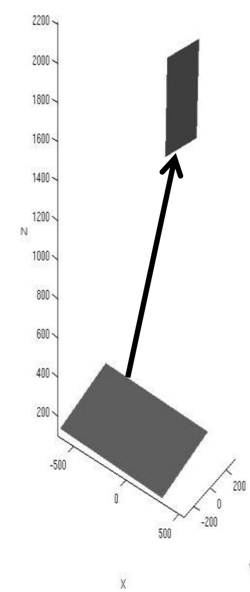
  determine right away that there is no collision
- On the other hand, if $d(R(q_A),W)+d(R(q_B),W)$ is small and $l(R(q_A), R(q_B))$ is large then the robot is moving close to obstacles and the trajectory must be broken down into small segments (like testing for collision)

Distance Between Two Objects



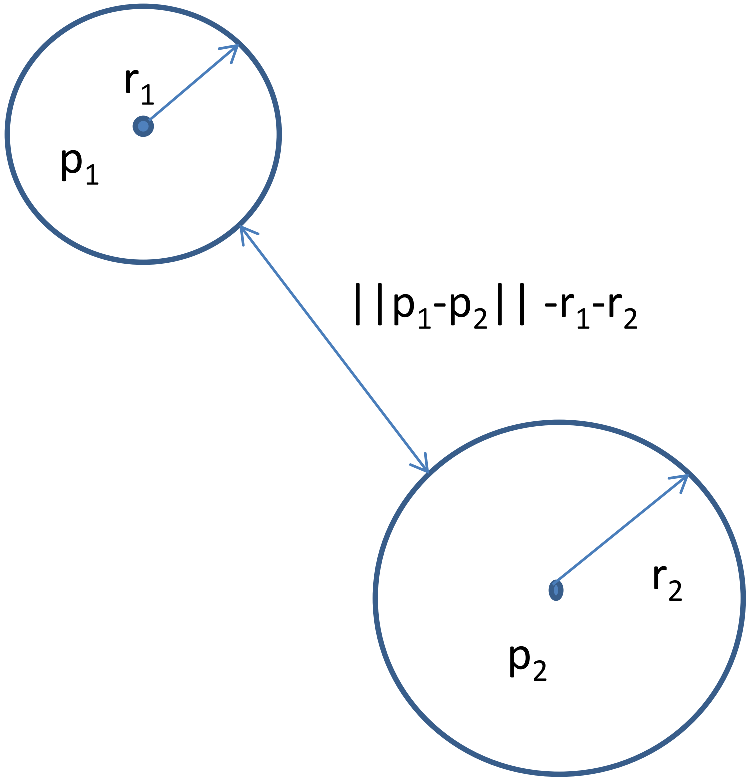
Larsen UNC 1999

- Use a hierarchy of swept sphere volumes (SSV)
  - Point Swept Volume
  - Line Swept Volume
  - Rectangular Swept Volume



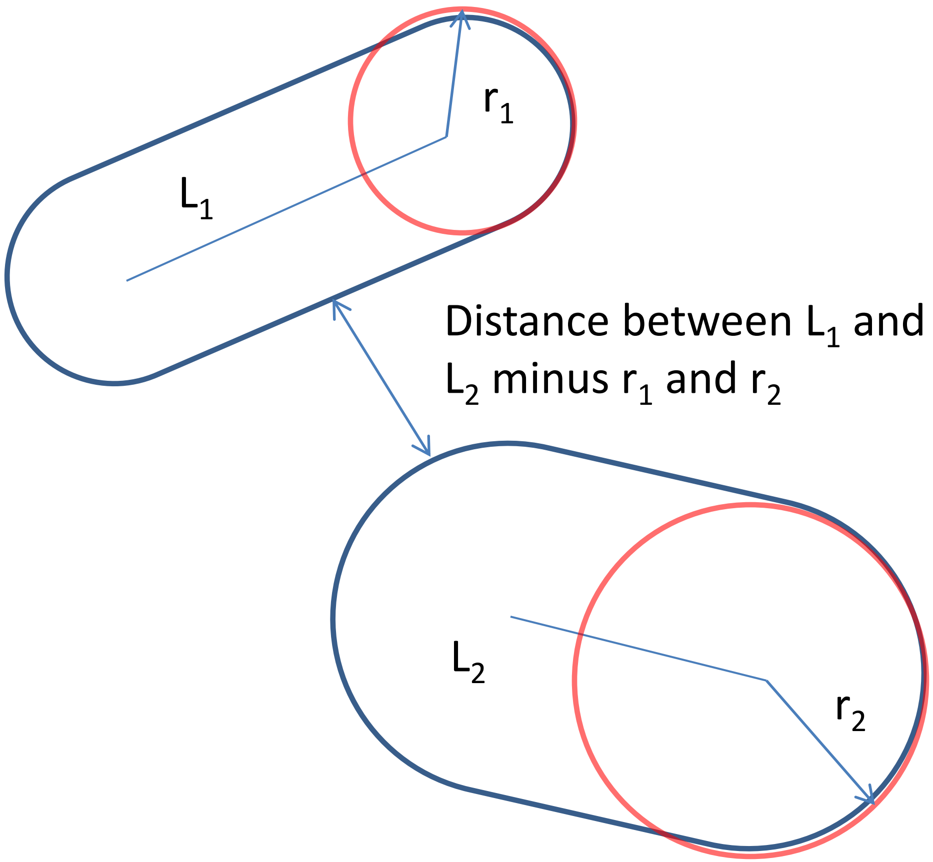
Point Swept Sphere

- Computing the distance between two 3D points is easy ($d = || p_1 - p_2 ||$)
- If you "sweep" each point with a sphere of radius r_1 and r_2, each point becomes a sphere of radius r_1 and r_2 respectively
- Computing the distance between two spheres is easy($d = || p_1 - p_2 || - r_1 - r_2$)



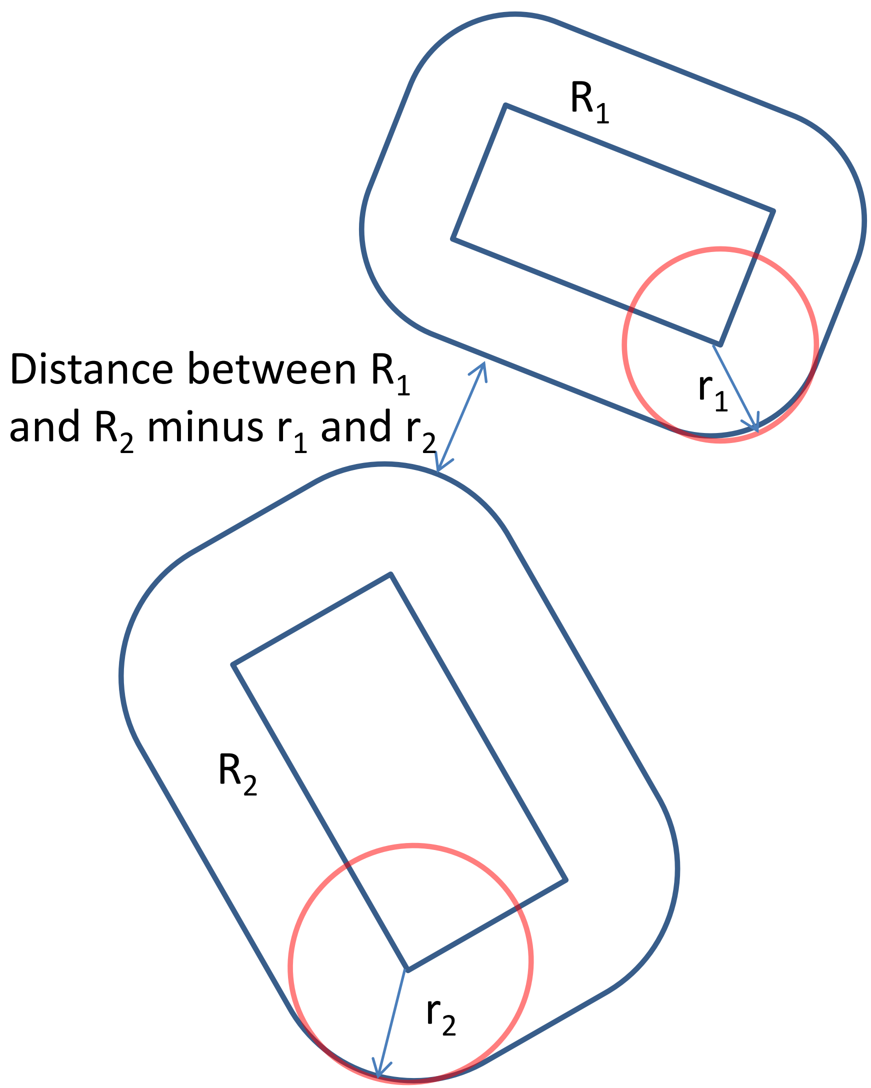
Line Swept Sphere (LSS)

- Computing the distance between two line segments L_1 and L_2 is "easy"
- If you "sweep" each line with a sphere of radius r_1 and r_2, each line expands by a sphere or radius r_1 and r_2 respectively
- Computing the distance between two LSS is the distance between both segments minus r_1 and r_2



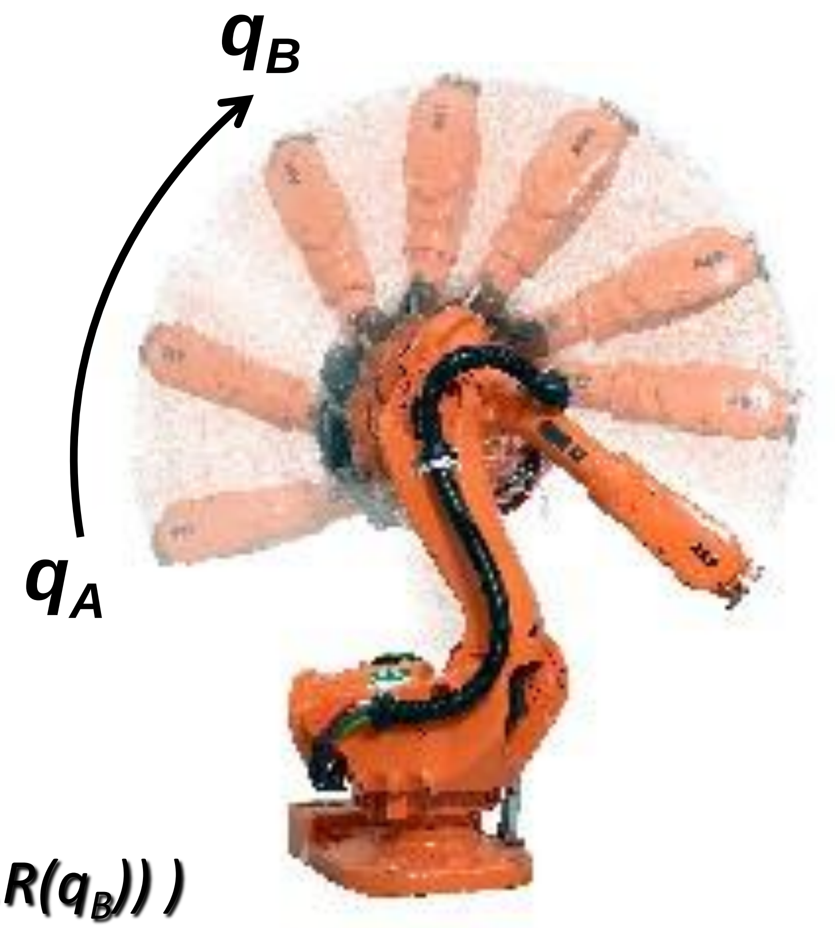
Rectangle Swept Sphere (RSS)

- Computing the distance between two rectangles R_1 and R_2 is "easy"
- If you "sweep" each rectangle with a sphere of radius r_1 and r_2, each rectangle expands by a sphere of radius r_1 and r_2 respectively
- Computing the distance between two RSS is the distance between both rectangles minus r_1 and r_2



Greatest Distance Traveled

- What is the point on the body's surface that travels the greatest distance from q_A to q_B?
- Upper bound the length of the trajectory traveled by any point on the volume between configuration q_A and q_B



$$d(R(q_A), W) + d(R(q_B), W) > O(\, l(R(q_A), R(q_B)) \,)$$

Upper Bound on $l(R(q_A), R(q_B))$

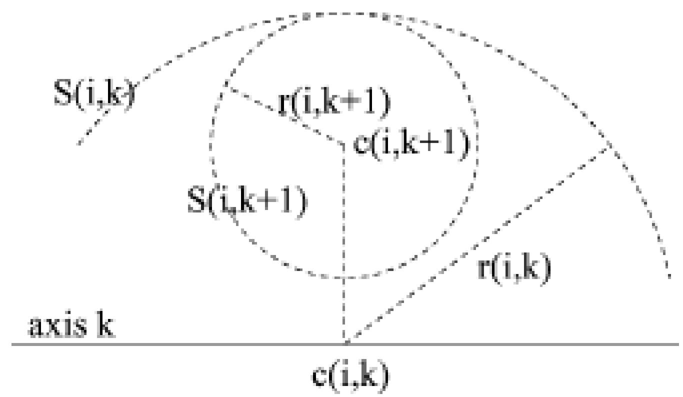
- What is the maximum contribution of each joint to $l(R(q_A), R(q_B))$?
  - Rotate the 3D model of each link by 360^o and fit an enclosing sphere and to the data point
  - The radius of the sphere guarantees that no point on the robot will move outside the spheres
  - For a rotation Δq_i of joint i, no point will travel a distance greater than $r_i \Delta q_i$ because of joint i.

Bound the Distance Travelled by Any Point on a Robot

Algorithm COMPUTE-SPHERE(i, k)

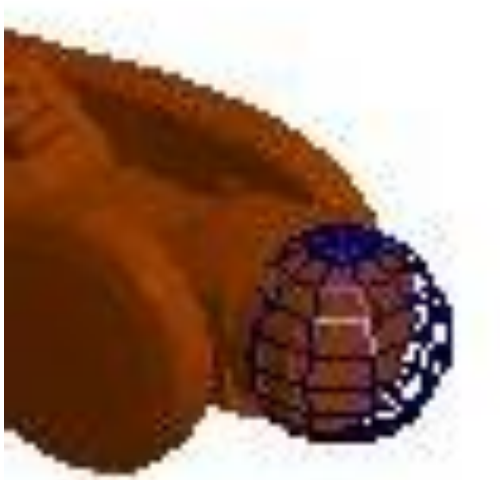
---

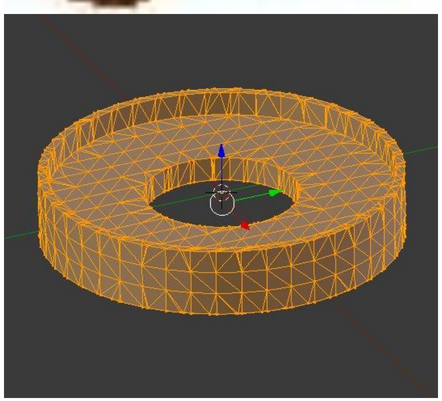
1. If $i = k$ then $S(i, k+1) \leftarrow$ ENCLOSING-SPHERE$(\mathcal{A}_i)$
2. Else $S(i, k+1) \leftarrow$ COMPUTE-SPHERE$(i, k+1)$
3. If joint k is prismatic then
   Sweep $S(i, k+1)$ along the full translational range of joint k and construct the sphere $S(i, k)$ that tightly encloses the swept volume.
4. Else if joint k is revolute then
   Sweep $S(i, k+1)$ around the axis of joint k by performing a full 2π rotation and construct the sphere $S(i, k)$ that tightly encloses the swept volume.
5. Return $S(i, k)$

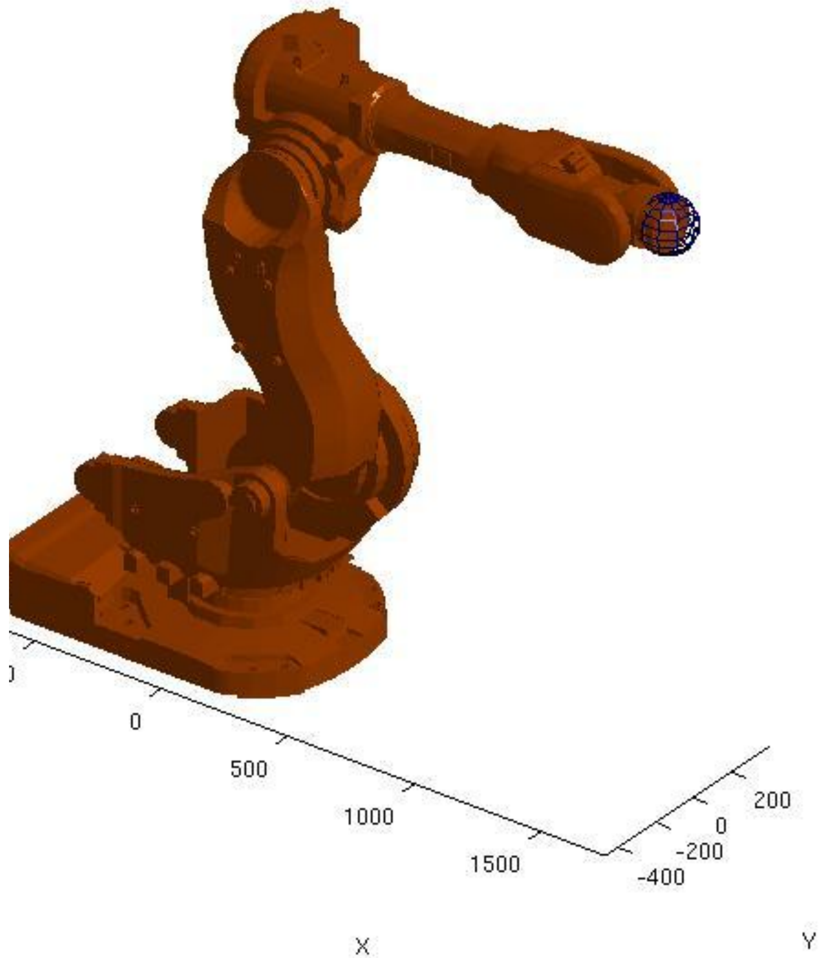


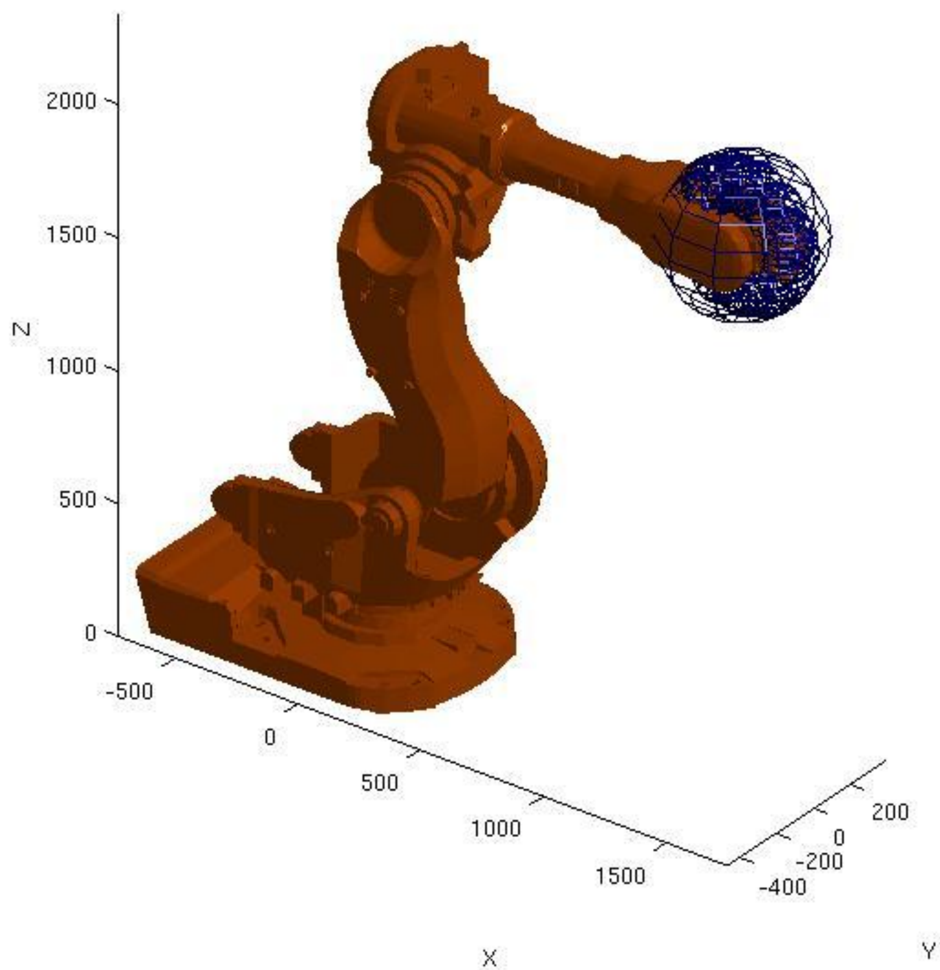
Schwarzer 2005

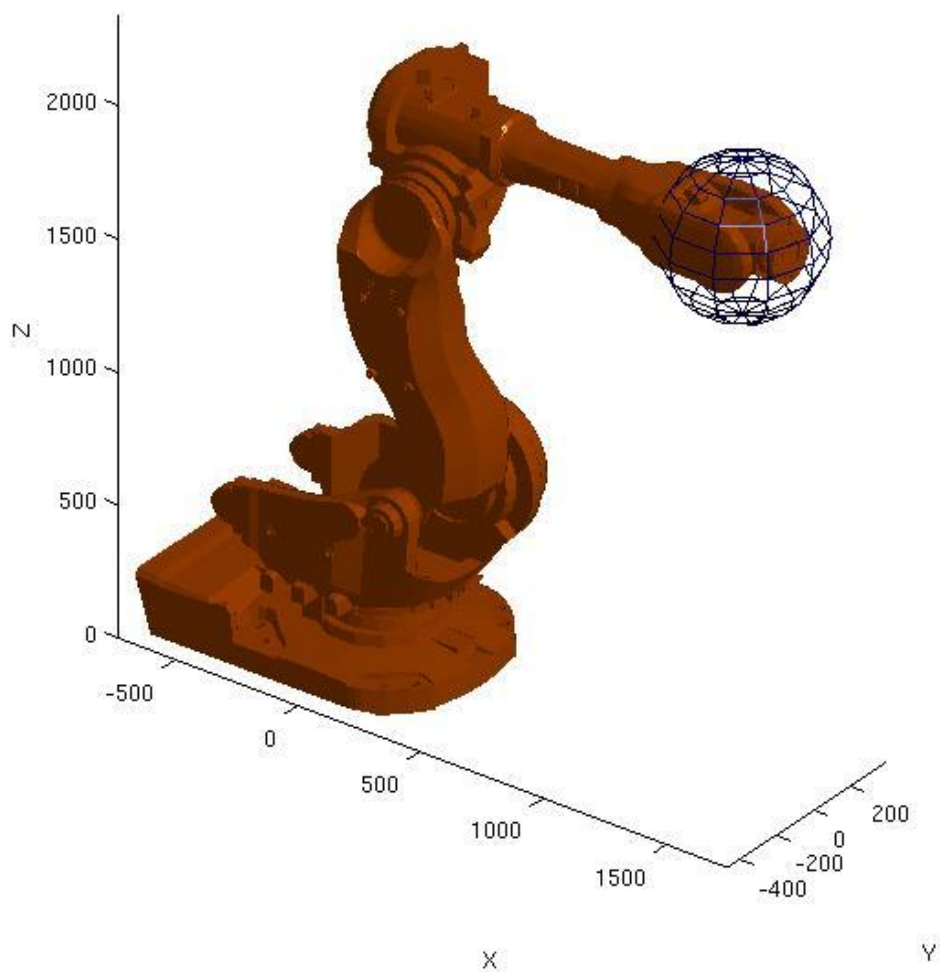
$$l(\, R(q_A), R(q_B)\,) = \sum_{k=1}^{i} R_k^i |q_{b,k} - q_{a,k}|$$



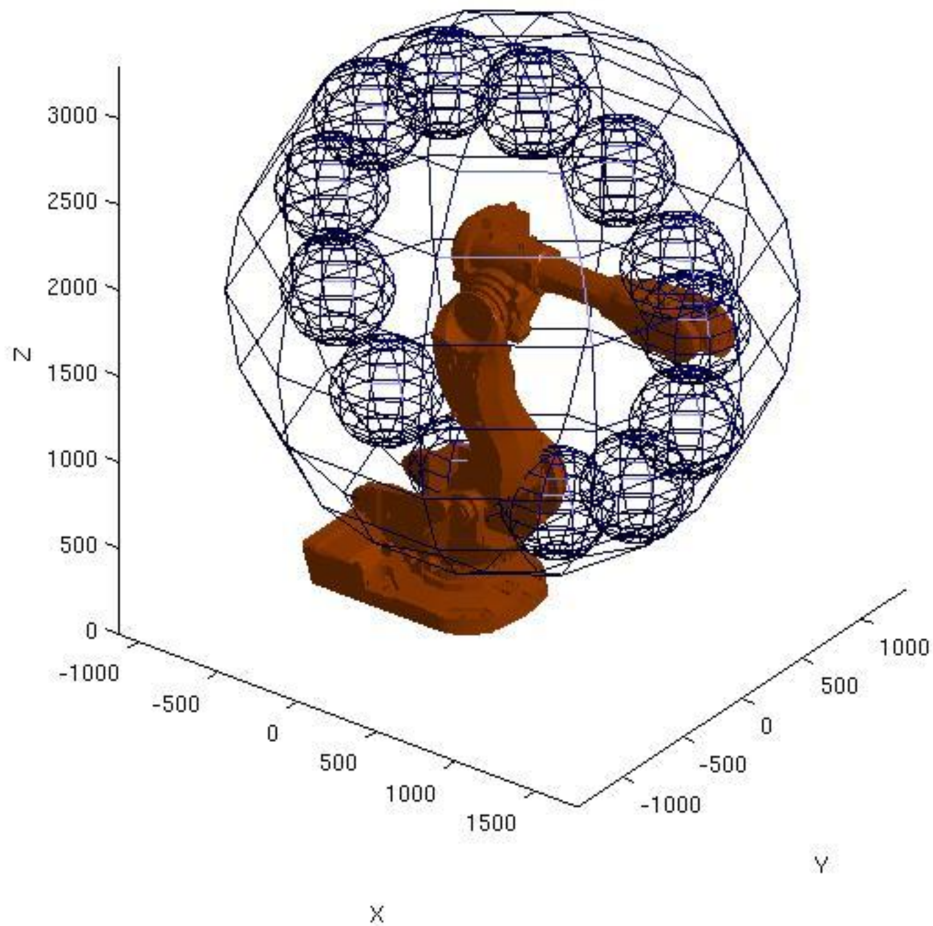



0
500
1000
1500
X
200
0
-200
-400
Y


2000
1500
1000
500
0
Z
-500
0
500
1000
1500
X
200
0
-200
-400
Y



2000
1500
Z
1000
500
0
-500
0
500
1000
1500
X
200
0
-200
-400
Y


3000
2500
2000
Z
1500
1000
500
0
-1000
-500
0
500
1000
1500
X
1000
500
0
-500
-1000
Y

Upper Bound on $l(R(q_A), R(q_B))$

- Given a trajectory between q_A and q_B
- Given a set of sphere radius r_i

$$\Delta q = | \ q_B - q_A \ |$$

$$l(R(q_A), R(q_B) < \Delta q_1 r_1 + \ldots + \Delta q_n r_n$$