LaValle_Ch5_3
Thursday, April 09, 2009
12:32 PM

5.3 Collision Detection

In most (geometric) M.P. problems, most computation time is spent checking sampled configurations for collisions.

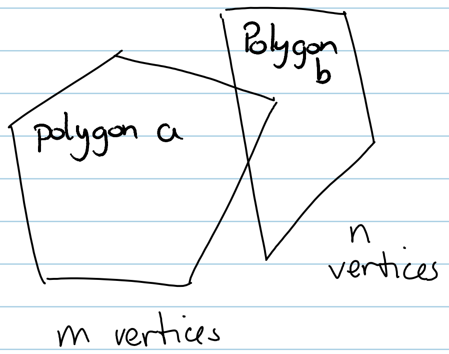
This is not true for dynamic systems with complex constraints.

Planar Case:

In the case of a convex polygonal robot & obstacles, collision checking can be done in linear time in the geometric complexity (i.e., the # of edge-vertex pairs).

a bit misleading terminology

geometric complexity of two convex polygons is $2mn$



Recall: Collision checking approach of section 4.3 is $O(mn)$.

Spatial Case:

Straight forward extension to 3D polyhedra is again linear in the # of geom. primitives. The # of geometric primitives is:

$$F_a V_b + E_a E_b + V_a F_b ,$$

where F_a, E_a, V_a are the #'s of Faces, Edges, and vertices of polyhedron "a", and F_b, E_b, V_b are #'s for polyhedron "b".

Collision checking

We just want a boolean result: yes, there is collision
or no, there is not

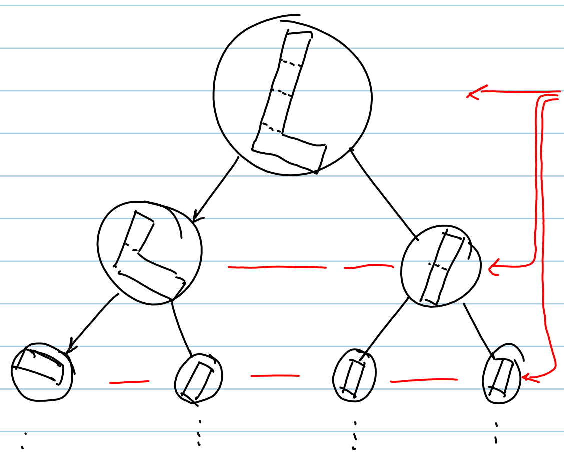
Use hierarchical tree representation of objects.

- root node is whole body
- leaf nodes are subsets w/ limited # of geom prims.
- other nodes are subsets of bodies
- each node's geometry is bounded by a simpler geom.
  - bounding sphere
  - axis-aligned bounding box (AABB)
  - oriented bounding box (OBB)
  - convex hull

fastest

indep. of details of object geom

Example using spheres (in 2D):

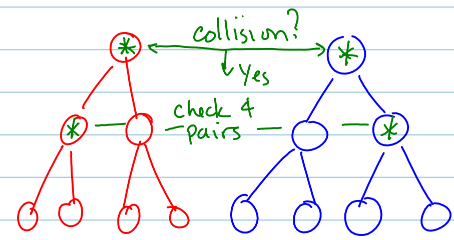


union of each row is complete object

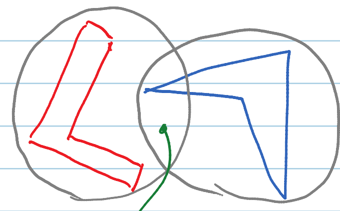
You could keep going until each leaf contains one convex polygon
but you might stop with n triangles or geometric

entities per leaf

Now consider collision checking with two bodies (two trees):



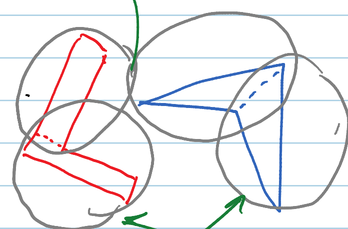
1. Root level test:
   Suppose the root nodes of E & F don't intersect. Then bodies E & F do not intersect.
   Done.



overlapping, so descend tree

2. If E & F roots intersect, then descend the trees one level

overlap again, so descend again.

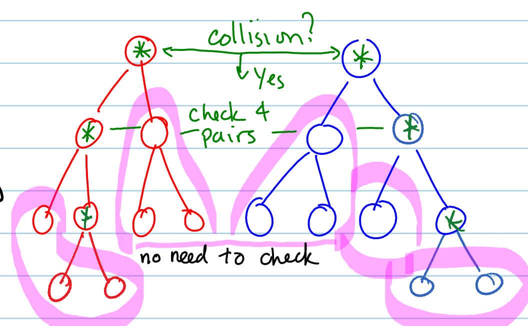


no overlap, so don't consider their children.

3. Continue recursively until no collisions occur at one level OR we reach the leaves.

4. At the leaves, collision check with geometric primitives

- Prune if bounding volumes don't overlap
- Might not have to descend to the leaves



Main advantage:
When bodies are far apart, very little work needs to be done.

What situations will not benefit from this decomposition?

Tight-fitting parts:

Medical catheterizations, assemblies

Other questions:

How do you balance the tree?

How do you decompose the object? most quickly shrinking bounding volume?

Should the tree be binary?

Is hierarchical geometric decomposition useful in distance comp?

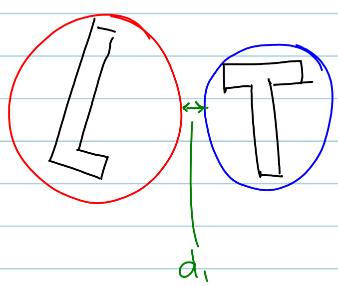
Why do I care about the last question?

Dynamics! We need ψ_n^l if $\psi_n^l <$ tolerance including $\psi_n^l < 0$ to form timestepping subproblems.

Modifications for distance computations. Does straight-forward extension work?

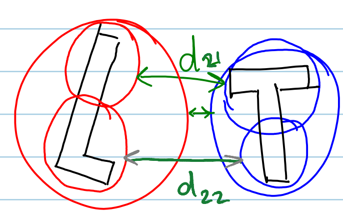
1. Compute distance between root bounding geometries first

Exact distance must be greater than or equal to d_1.



2. Descend to second level.

Compute distances between pairs of subsets $\Rightarrow d_2$

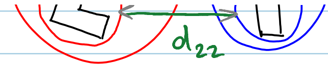


Let

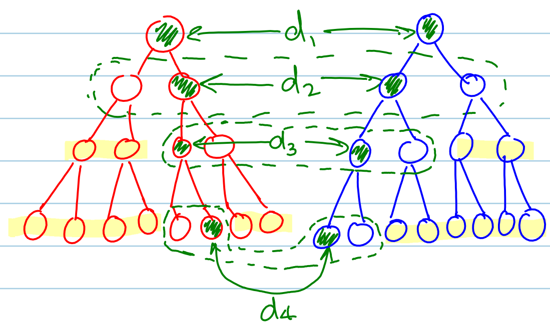
$d_2 = \min(d_{21}, d_{22})$

Can we prune

pairs of subsets $\rightarrow d_2$



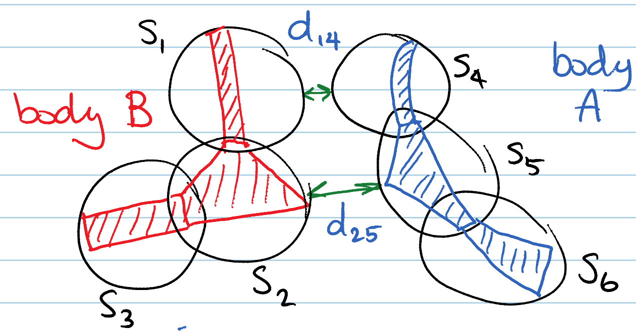
Can we prune like this?



- Unlike the problem of C.D., to obtain distance we <u>must</u> recurse to the leaves!
- Expensive distance computations must be performed!
- Pruning is <u>NOT</u> as simple!

Actual distance is d_{25}

$d_{25} > d_{14}$, so can't prune S_2, S_3, S_5, S_6



Possible condition for pruning with bounding spheres:

Let r_i be radius of i^{th} bounding sphere.

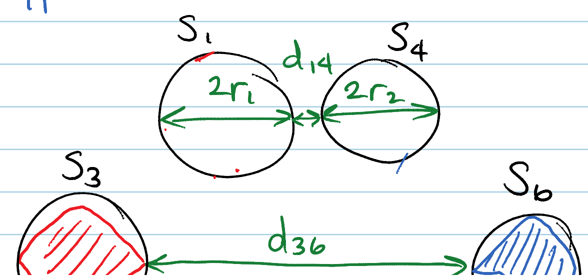
Let d_{ij} be distance between S_i & S_j i on A, j on B

Let d^*_{ij} be smallest distance found so far

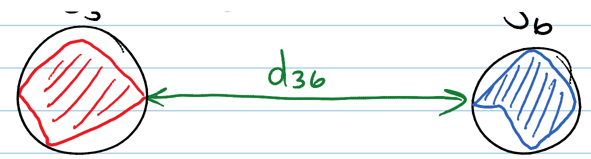
You can prune S_k & S_l if

$d_{kl} > d^*_{ij} + 2r_i + 2r_j$

$d_{36} > d_{14} + 2(r_1 + r_2)$

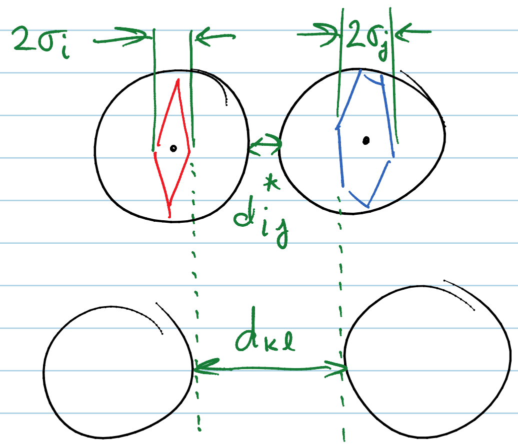


$d_{36} > d_{14} + 2(r_1 + r_2)$



If decomposition is all convex parts and bounding volume is smallest sphere, then pruning can be more aggressive: prune if $d_{kl} > d^*_{ij} + r_i + r_j$

If you also knew minimum dimension of parts you could bound more tightly,



prune if something like

$$d_{kl} > d^*_{ij} + r_i + r_j - \sigma_i - \sigma_j$$

For state of the art in distance computation, see work of Dinesh Manocha at Univ. of North Carolina.

Open question: How can one most efficiently obtain all feature pairs within a given distance (and their distances (including penetration depths))?

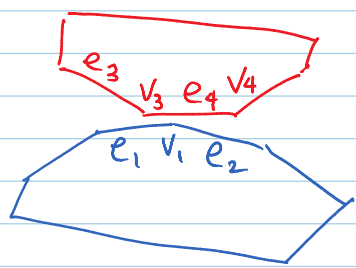
In multibody dynamics, we need to know:

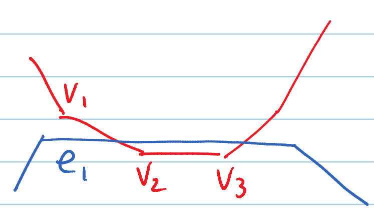
$dist(e_1, v_3) < \epsilon$

$dist(v_1, e_4) < \epsilon$

$dist(v_4, e_2) < \epsilon$

so we can construct φ'_n correctly and $\therefore$ simulate accurately.





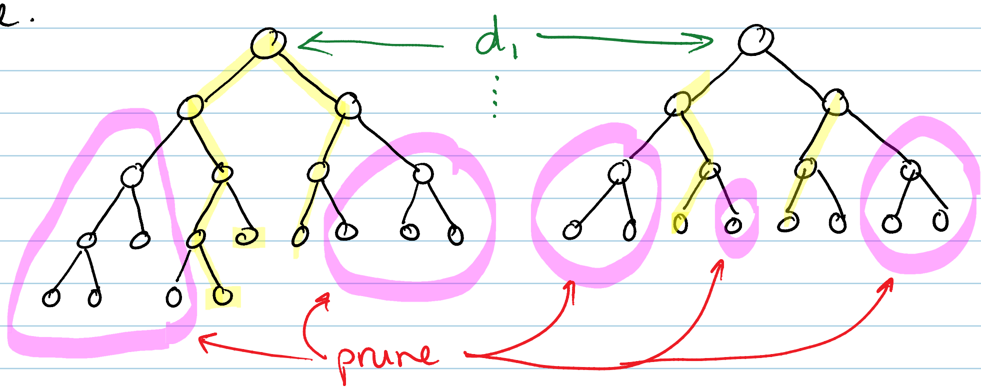
and ∴ simulate accurately.

here we need $(v_1, e_1), (v_2, e_1), (v_3, e_1)$

Modify Pruning condition

if $d_{ke} > d^*_{ij} + r_i + r_j - \sigma_i - \sigma_j + \epsilon$

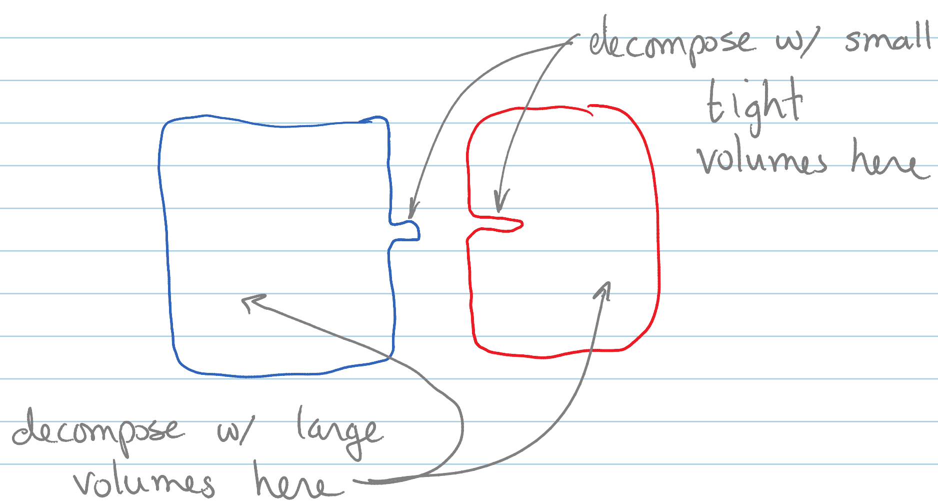
then prune.



Note:

For both collision detection & distance computation, the specific hierarchical decomposition affects computation time

For example if we know all the "action" is in certain regions, put small bounds on those regions



Incremental Collision Checking

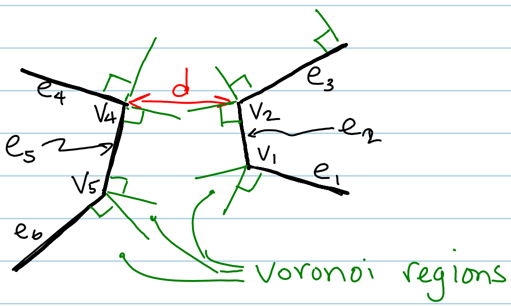
Use spatial coherence to speed collision checking.

Can achieve almost constant time.

Need knowledge of topology for this to work.

Triangle or polygon soup not good rep. for this.

Consider case of convex polygons in the plane



Voronoi region is region closest to a geometric feature. A point in a region is closest to the corresp. feature of a convex polygon.

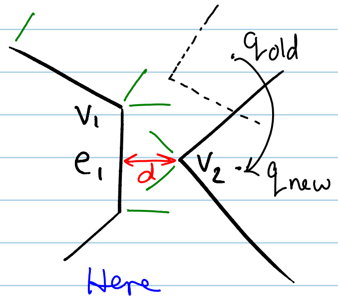
Polygon distance condition

Let F_1 & F_2 be geometric features of polygons 1 & 2, respectively, where geom. features are edges and vertices.

Let $(x_1, y_1) \in F_1$ and $(x_2, y_2) \in F_2$ be the closest pair of points on F_1 & F_2 among all points on F_1 & F_2

If $(x_1, y_1) \in Vor(F_2)$ and $(x_2, y_2) \in Vor(F_1)$, then the distance between (x_1, y_1) and (x_2, y_2) is the distance between the polygons.

Incremental collision checking uses knowledge of topology (i.e. connectivity of edges and vertices) to determine which feature pair is which feature pair is likely to contain the closest points



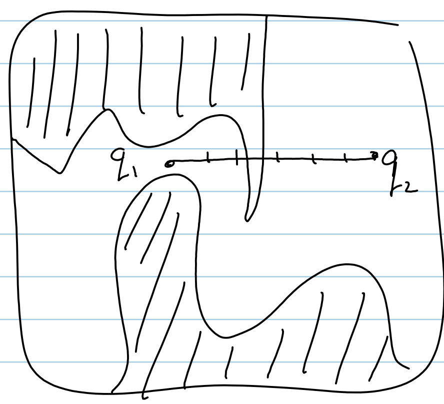
$\boxed{v_1, v_2} \xrightarrow{\Delta q} \boxed{e_1, v_2}$

Here minimum distance shifted by one Voronoi region on one polygon.

The basic idea is that if the polygon's relative config changes only slightly, then one can quickly find the closest feature pair from the previously found pair. For example, the search should require at most moving up the tree a small # of links.

Collision checking along path segments

Sample path checking for collision at each point.



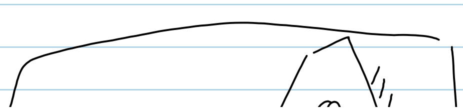
Could use multi-resolution scheme.

If collision found, then discard path segment

How can we guarantee that a finite path segment is 100% collision free?

Derive bounds. Unfortunately they tend to be loose.

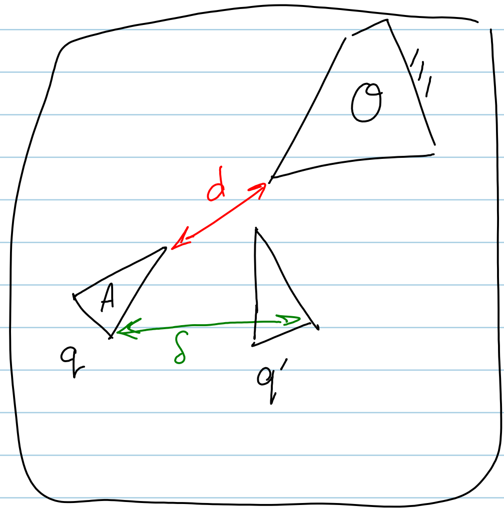
Planar Case



Planar Case

Let d be the distance at q

Let δ be the max. displacement of any point on A while moving to q'.



If $\delta < d$, is collision possible? Yes! Since part may move in strange way.

However if δ denotes the longest path taken by any $a \in A$ while moving from q to q', then if $\delta < d$, the segment from q to q' is collision free.

Let the config of A be denoted by $(x_t, y_t) \in \mathbb{R}^2$, $\theta \in S^1$

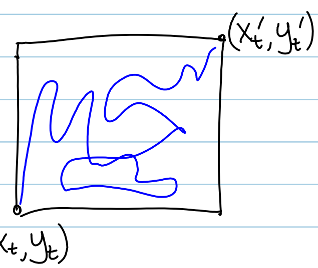
Translation

If A translates from (x_t, y_t) to (x_t', y_t'), then

$$\delta = \sqrt{(x_t - x_t')^2 + (y_t - y_t')^2} \quad , \forall a \in A$$

Suppose we only require path to stay in a box

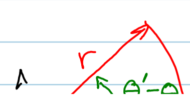
Then max δ of $a \in A$ is bounded by



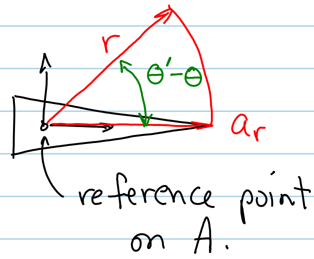
$$\delta \leq |x_t - x_t'| + |y_t - y_t'| \quad \forall a \in A$$

Suppose we include rotation from θ to θ' along the shortest path is S'.

Let a be point further



Let a_r be point furthers from reference point, and let r be its distance



The greatest displacement any point $a \in A$ can experience is bounded by $r|\theta' - \theta|$

If position & orientation may vary between their limits, then a collision free region of C_{free} is:

$$\text{subset of } C_{free} = \{(x_t', y_t', \theta') \in C \mid |x_t - x_t'| + \dots$$

$$\dots + |y_t - y_t'| + r|\theta - \theta'| < d\}$$

The bound can be used to choose a collision-checking stepsize Δq such that no collisions will be missed along the path.

An analogous approach can be used in 3D worlds.

When the robot has multiple links, the approach becomes strongly configuration dependent, making it difficult to apply cost-effectively.