LaValle Ch5.2 Sampling Theory

2/19/18

The number of points in $\mathcal{C}$ (or X) is uncountably infinite, yet even if our alg runs forever, the space will be probed with only a countable # of samples. And, in practice, only a finite # of samples can be used.

∴ samples must be drawn "carefully".

It is important to distinguish between the sample set and the sequence (the order in which they are drawn).

LaValle_Ch5-2

Tuesday, April 10, 2012
9:28 PM

Sampling Theory

Connect to C-space of arb. dimension

Bring the uniform random quat gen. here too

End with how to sample $S^l \times I^m \times SO(n)$

Then do same in deterministic sampling.

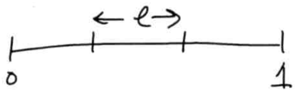
Random Sampling

Motivate this better

$U[0,1]$ random sampling is dense on $[0,1]$.

Suppose we pick k points at random from $U[0,1]$.

Pick any interval of length e.



What is the prob. that none of the k points falls in the interval?

For one point, $\text{prob} = 1-e$

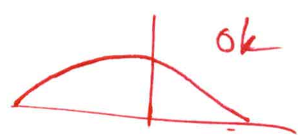
Assuming the points were drawn independently, then

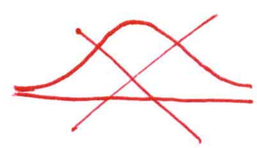
$$\text{prob} = (1-e)^k$$

Look at limit as $k \to \infty$

$$\lim_{k\to\infty} (1-e)^k = 0 \qquad \forall e \mid 0 < e \leq 1$$

Important implication. The infinite set of points drawn at random from $U[0,1]$ is dense on $[0,1]$ w/ probability 1 as the # of points drawn $\to \infty$.





Using the same approach, one can see that iid. samples from any distribution with finite tails (finite support) will be dense on the support interval as the # of samples goes to ∞.

Note: Random sampling via computer is "pseudo-random". It is actually deterministic & periodic.

Show page 201 LaValle:

Random sampling may take long time to explore some areas of C (or X).

Low-Dispersion Sampling

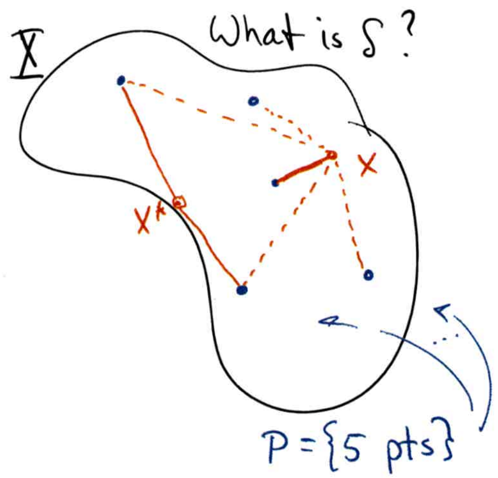
An alternative to random sampling

Idea – sample to make largest contiguous uncovered area as small as possible.

Def: Dispersion δ of a finite set P of samples in a metric space (X, ρ) is:

$$\delta(P) = \max_{x \in X}\left\{ \min_{p \in P} \left\{ \rho(x, p) \right\} \right\}$$

A good goal for sampling of C (or X) is to minimize $\delta(P)$. Why? It implies equal good coverage of the space.



Consider one pt, $x \in X$.

What is $\min_{p \in P}\left(\rho(x, p)\right)$?

Next, move x to find the largest minimum distance.

How to place points?

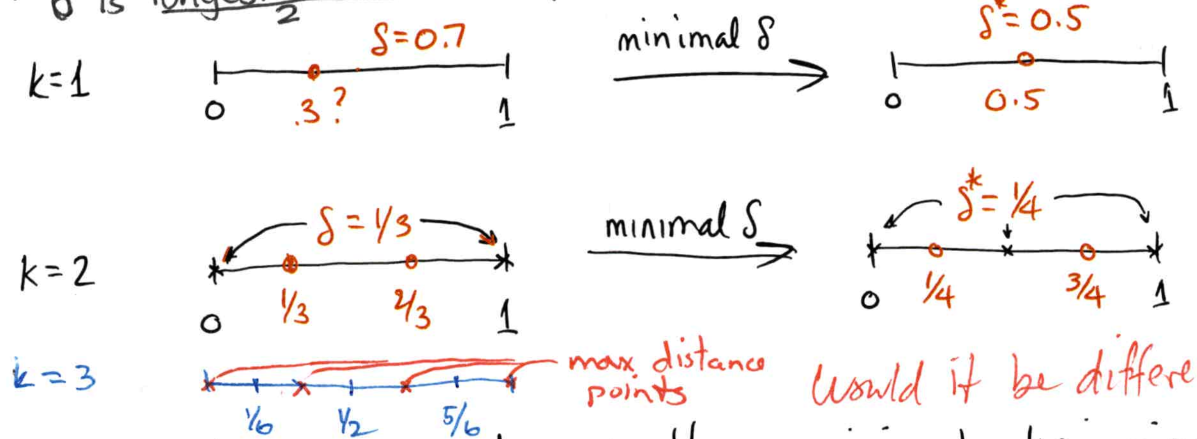
Perhaps the next $p \in P$ should be the x that gave $\delta(P)$!

Case 1 # of pts known

Consider sampling I' with k samples:

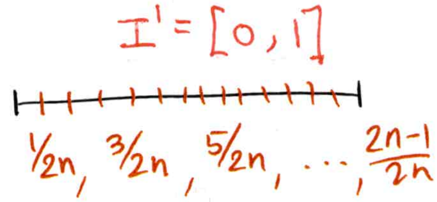
* δ is* longest interval w/o points from P.
2

If any point in optimal placement is moved, then $\delta \uparrow$.



Would it be different in S^1? ~~No~~ just rotate

If k is known in advance, then minimal dispersion is achieved by

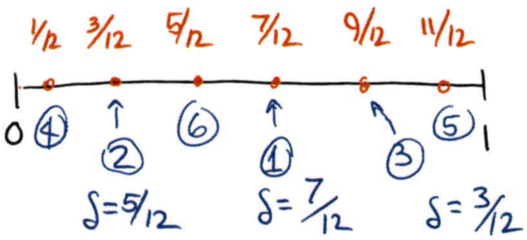


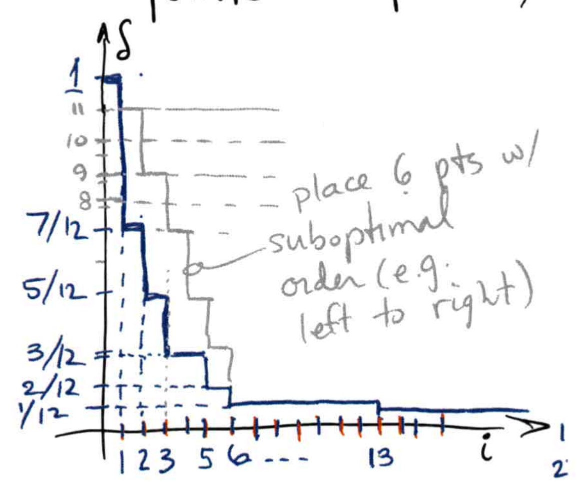
$\delta^* = \frac{1}{2n}$

The idea is to use the point placements in sequence while you build graph.

What's a "good" order of placement? (maybe you try to solve problem before all points are placed)

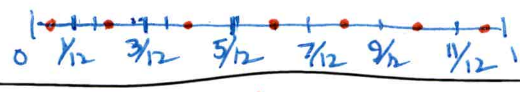
If low dispersion is good, then bring it down as quickly as possible. e.g. $k=6 \Rightarrow \delta^* = 1/12$





If $\delta = 3/12$ is enough resolution to solve problem, then $1/2$ the work was saved by using a good placement order.

Note: must add at least 7 points in example above to reduce the dispersion.
Number grows exponentially w/ Dim (C) !
Dim (X)



Move to 2D.

All points in box have ~~dispersion~~ L_∞

$L_\infty \triangleq \max_i (\|\Delta x_i\|)$

$\delta = r$

Less than r.
Find place for box w/ no pts inside.
Grow box

$\delta = r$

placing a pt in circle will reduce δ (by as much as $1/2$).

points of greatest L_∞ distance from any sample pt.

point greatest L_2 distance from any sample point

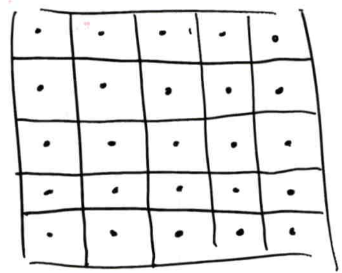
Circle will always touch three points in P.

Notes: In these two examples, adding just 1 pt. can reduce the dispersion, but the dispersion is not optimal (minimal) for the given # of points.

reduce the dispersion, but the dispersion is not optimal.

Optimal L_∞ dispersion for a given # of pts is known to be obtained by a grid with sample pts at the centers (a.k.a. a Sukharev grid)

e.g. 5x5 Sukharev grid on I^2 →



How big are the cubes in higher dimensions?
i.e. What's the optimal L_∞ dispersion for k pts in I^n ?

How many cubes in one axis direction in $I^n = \{(0,1) \times \cdots \times (0,1)\}$
if using k points?

$\lfloor k^{1/n} \rfloor$ floor function

2D

$\lfloor 25^{\frac{1}{2}} \rfloor = 5$ cubes

$\lfloor 29^{1/2} \rfloor = 5$

⋮

$\lfloor 36^{1/2} \rfloor = 6$

⋮

3D

$\lfloor 125^{1/3} \rfloor = 5$

$\lfloor 216^{1/3} \rfloor = 6$

$\lfloor 343^{1/3} \rfloor = 7$

⋮

For a unit cube in $\mathbb{R}^n$: → $\delta = \frac{1}{2} \frac{1}{\lfloor k^{1/n} \rfloor}$

There are $k - (\lfloor k^{1/n} \rfloor)^n$ extra points.
These can be scattered anywhere w/o affecting the dispersion. - but place at points of higher resolution grid to reduce dispersion elsewhere?

no. grid pts of 6x6 don't line up w/ 5x5

If you don't know k in advance, then you don't know optimal cube size.

Problem: If you want to maintain a given value of dispersion as the dimension of the space grows, then the # of sample points will grow exponentially.

Consider case where there are no extra pts.

$$\delta = \frac{1}{2\lfloor k^{1/\cancel{d}\,n} \rfloor} = \cancel{\text{constant}}\ c \Rightarrow k^{1/\cancel{d}\,n} = \cancel{\text{constant}}\ c$$

$$\cancel{\tfrac{1}{d}\log k = \log c \Rightarrow \log k = d \log c \Rightarrow \boxed{k = c^{d}}}$$

$\delta = \frac{1}{2k^{1/n}} = c$

$$\Rightarrow k^{-1/n} = 2c \Rightarrow -\frac{1}{n}\log(k) = \log(2c) \Rightarrow \log(k) = -n\log(2c) \Rightarrow k = \left(\frac{1}{2c}\right)^n$$

A nice thing about grids is that it is easy to determine neighboring points (via generating vectors).

i.e. They have a known lattice structure.

If resolution needed is not known in advance, then neither is k!

What if # of pts k is not known in advance?

Best strategy may be iterative doubling of resolution of Sukharev grid.

pts increases by factor of 2^n for each doubling! Accepts partial grids.

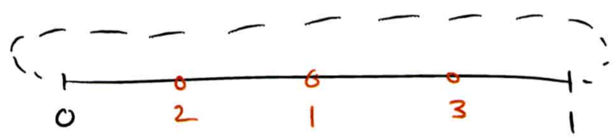
At $n=10$ full grids appear at $1, 2^{10}, 2^{20}, 2^{30}$ .... Impractical

Use Van der Corput sequences for sampling order.

Consider $I'/\sim\ = S'$

δ is optimal

$$\delta = 2^{-(\lfloor \log_2(k) \rfloor + 1)}$$



$k = 1 \rightarrow \delta = 1/2$

If target resolution is known, compute k.
If max k is known, compute dispersion

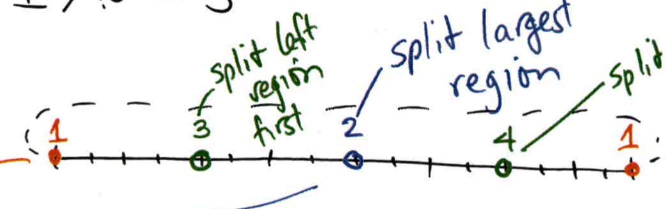
$k = 2$ or $3 \longrightarrow \delta = 1/4$

$k = 4,5,6,7 \longrightarrow \delta = 1/8$

$k = 8, \dots, 15 \longrightarrow \delta = 1/16$

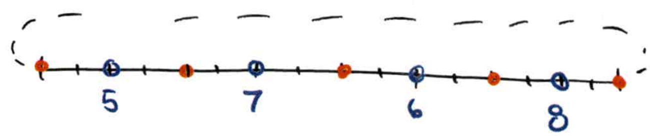
Van der Corput sequence

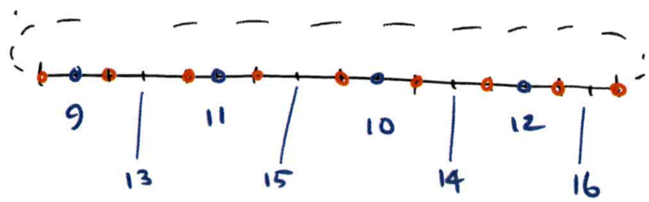
Suppose we will place up to 16 pts in $I^1/\sim\ = S^1$



Naive ordering	Binary	Reverse Binary	
0	0.0000	0.0000	0~1
1/16	0.0001	0.1000	1/2
2/16	0.0010	0.0100	1/4
3/16	0.0011	0.1100	3/4
⋮	⋮	⋮	
14/16	0.1110	0.0111	
15/16	0.1111	0.1111	

This is like B-F search





Suppose you choose 2^n pts, then want more samples:

n=1

0.0
0.1

2 copies

n=2

0.00
0.10
0.01
0.11

$\Longrightarrow$

n=3

0.000
0.100
0.010
0.110

0.001
0.101
0.011
0.111

add 1/8 to every point

$\Longrightarrow$

n=4

0.0000
0.1000
0.0100
0.1100
0.0010
0.1010
0.0110

add 1/16

0.0001
0.1001
0.0101
0.1101
0.0011
0.1011

0.00
0.10
0.01
0.11

$\Rightarrow$

0.000
0.100
0.010
0.110

0.001
0.101
0.011
0.111

add 1/8 to all

$\Rightarrow$

0.0000
0.1000
0.0100
0.1100
0.0010
0.1010
0.0110
0.1110

0.0001
0.1001
0.0101
0.1101
0.0011
0.1011
0.0111
0.1111

add 1/16 to all

Don't simply increase by one digit each time all pts have been tried. This forces you into building complete grids

Higher Dimensions

Van der Corput extends easily to I^n & S^n

<u>Not</u> easily to SO(3)

Lavalle suggests a good approximation for SO(3)

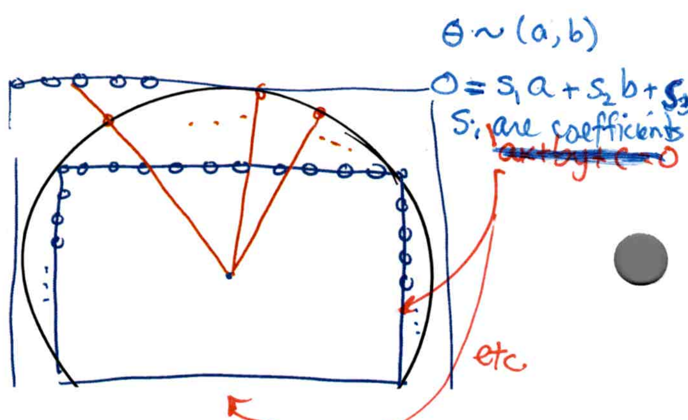
Circumscribe a 3D hyper cube on S^3 (recall, S^3 is space of unit quaternions)

Apply Van der Corput sampling to each face (I^3) and "lift" each sample to S^3

i.e. for each sample, replace it with $\frac{s}{\|s\|}$.

Analogy in $\mathbb{R}^2$

Pretend ~~Assume~~ Van der Corput sequence cannot be placed on S^1.



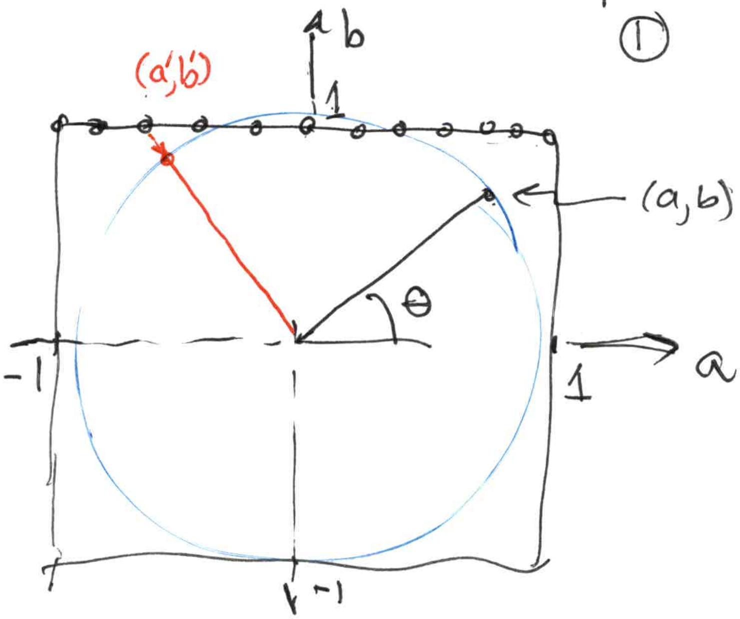
Represent orientation by (a,b), with $a^2+b^2=1$

2/20/18

1

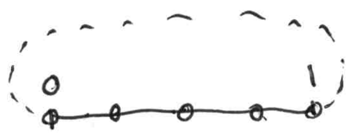
Enclose 1-sphere in square in $\mathbb{R}^2$

place Van der Corput sequence on each edge of square



Obtain (a,b)'s that approx uniformly sample S^1 by projection.

Take (a,b) from Van der Corput sequence and normalize



Multiply by 2 and subtract 1.

Then let $b'=1$ and $a' =$ VdC numbers.

Project by $\dfrac{(a',b')}{(a'^2+b'^2)^{1/2}} = (a,b)$

2/20/18

2

How to do this for $SO(3)$?

Use $h = (a, b, c, d)$, $a^2 + b^2 + c^2 + d^2 = 1$

Each face of hypercube in $\mathbb{R}^4$ is

$$s_1 a + s_2 b + s_3 c + s_4 d + s_5 = 0$$

There are 8 3D faces

faces $\perp$ a-axis: $s_1 = 1$, $s_2 = s_3 = s_4 = 0$, $s_5 = \pm 1$

similar for
b, c, d axes.

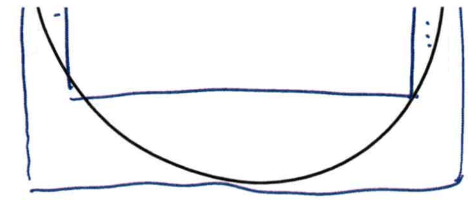
<u>Case for a-axis:</u>

Place grid of points (VdC (or other)) on hyper-cube

$$\{(b, c, d) \mid -1 \leq b \leq 1, -1 \leq c \leq 1, -1 \leq d \leq 1\}$$

Let $a = 1$ for all points in grid

Project (a, b, c, d) onto S^3 by normalizing.



How would you implement this for SO(3)? (Not in text)

$h = (a, b, c, d)$

Each face of the hypercube is:

Analogous to $ax+by+c=0$ defining a line in $\mathbb{R}^2$

$$s_1 a + s_2 b + s_3 c + s_4 d + s_5 = 0$$

where s_i are coefficients

The are 8 faces:

2 faces $\perp$ to a-axis $\quad s_1 = 1,\ s_2 = s_3 = s_4 = 0,\ s_5 = \pm 1$

$s_2 = 1,\ s_1 = s_3 = s_4 = 0 \quad s_5 = \pm 1$

$s_3 = 1,\ s_1 = s_2 = s_4 = 0 \quad s_5 = \pm 1$

$s_4 = 1,\ s_1 = s_2 = s_3 = 0 \quad s_5 = \pm 1$

Case $a = 1$:

Place grid on $\{(b, c, d) \mid -1 \le b \le 1,\ -1 \le c \le 1,\ -1 \le d \le 1\}$

Project (a, b, c, d) onto unit sphere by normalizing (a, b, c, d)

~~We will not cover low-discrepancy sampling~~

~~The motivation is to move points of coordinate axes, because unpredictably this can be bad.~~ See next page

We will not cover low discrepancy sampling

Main goal is to obtain uniform sampling w/o grid alignment.

Grid alignment seems to increase the variance of planning times – we prefer predictability!

Well-known low-discrp. sequences are:

Halton
Hammersley
} see Section 5.24 of Lavalle's text

Also google "van der corput fsu" to find open source code to generate Halton & Hammersley sequences in multi-dimensional spaces!

LaVallе 5.3: Collision Detection

2/21/18

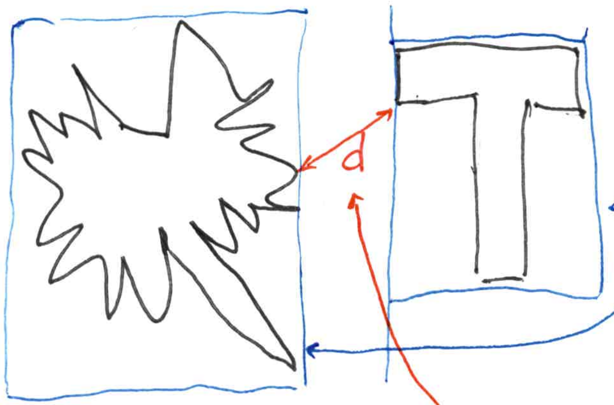
In most M.P. problems, most of the computation time is spent checking sampled configurations for collisions

∴ We should understand its inner workings.

Collision Detection ← faster { Useful for problems where contacts are not allowed.

vs.

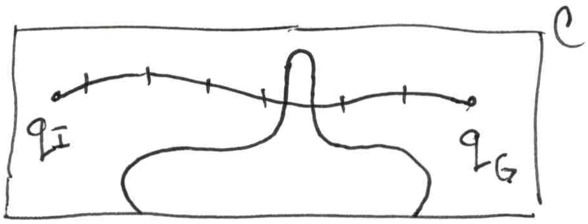
Distance Computation ← slower Useful when distances are controlled or contacts are needed (e.g., manipulation)



Sometimes simple bounding volumes suffice for C.D.

Distance computate requires some detailed geometric information of both bodies.

Sometimes pointwise C.D. is not enough



Continuous collision detection check path segment, not just pts.

Two Phase C.D.

Broad Phase N bodies Which pairs (out of $\frac{N(N-1)}{2}$) must be checked

\# of checks is $\boxed{O(N^2)}$

Narrow Phase M geometric features on each body

$\left.\begin{array}{l}\boxed{O(M^2)} \text{ detailed checks} \\ \text{(VF, FV, EE) to perform.}\end{array}\right\}$ per body pair

which can be avoided?

Brute-force approach is $O(M^2 N^2)$.

Two-phase C.D. aims to eliminate as many as possible of the collision checks (or distance computations) as possible

Broad Phase purpose – quickly prune many body pairs

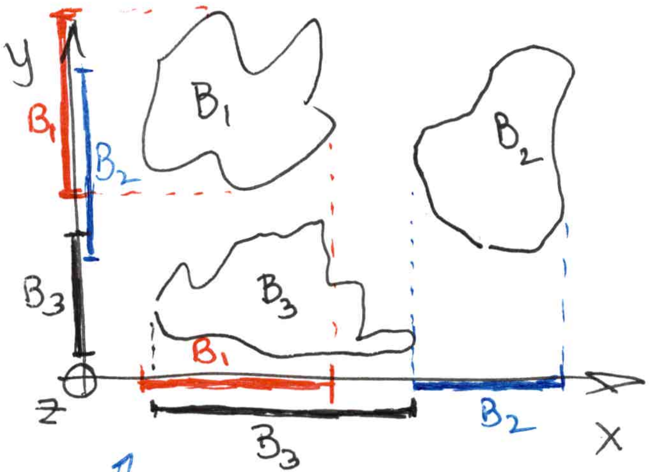
- Sweep & prune
- Bounding volumes
- Spatial subdivision
- Space-Time bounds

Output of Broad Phase is list of pairs of bodies to check.

Sweep & Prune

For each axis

- project body points onto axis.
- check for overlap. If none, remove from list of checks.



Example: On y-axis, B_1 & B_3 don't overlap. Don't check them for collision

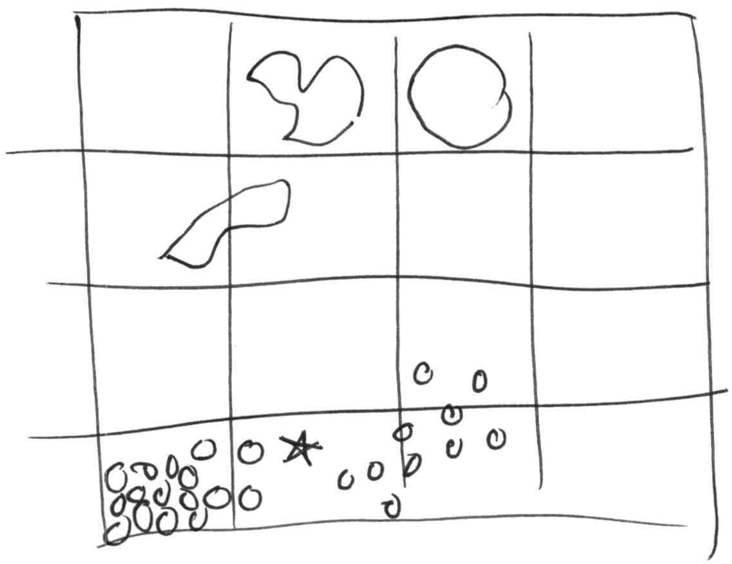
On x-axis all three ranges overlap, so no savings.

Note: In multibody simulation, you typically need distances between all pairs of bodies, suggesting that you need to do all $N(N-1)/2$ distance computations.

However, sometimes distance doesn't matter if $d > \bar{d}$, where $\bar{d}$ is some threshold. Then above alg is useful if you find bodies $> \bar{d}$ in any coordinate. Then by Pythagorean Thm, $d > \bar{d}$.

Spatial Subdivision

- Make grid of cells (voxels) larger than largest object

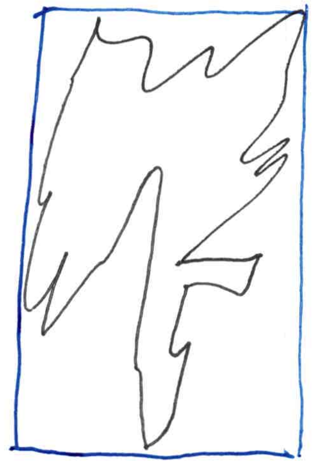


- keep list of all objects overlapping each cell

- Objects can't overlap if they don't overlap a common cell.

If objects have a large size ratio, Spatial subdivision may not help much.

Bounding Volumes

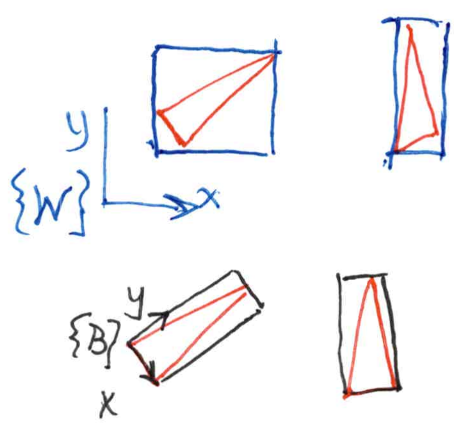


- Use geometrically object to contain a geometrically complex object.
  - sphere, box, convex hull (of polyhedron)
- Then apply sweep & prune to bounding volumes.
- If C.D. required, first compute with bounding volumes.

Choice of bounding ~~Vol~~ boxes:

AABB ← axis aligned

OBB ← oriented



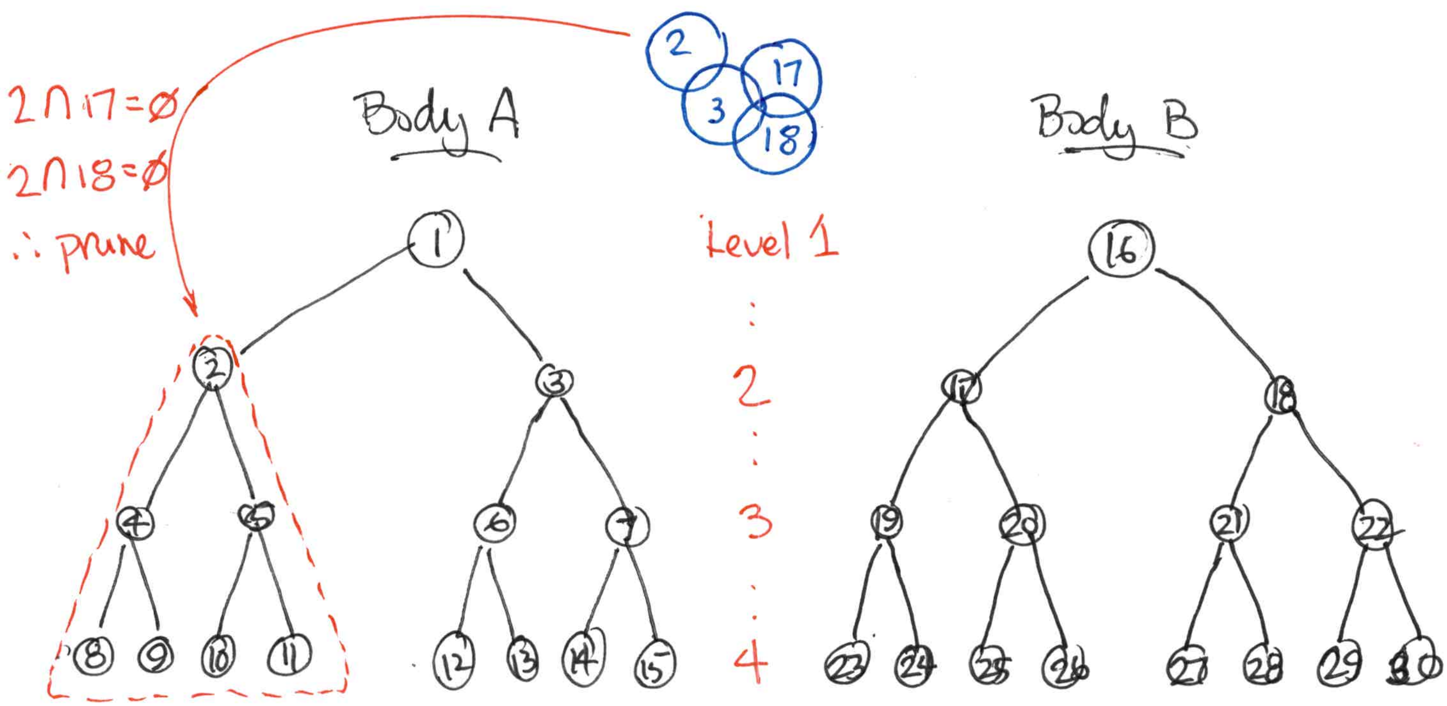
Recompute box after rotation (or use very big box)

Recompute box vertices after rotation

2/21/18

Narrow Phase

- Use hierarchy of bounding volumes to represent body geometry at varying levels of detail.
- Output of Narrow Phase is list of pairs of geometric primitives to check.
  (FV, VF, EE)



<u>C.D. Pruning Rule:</u> Node i of body A is pruned iff its volume intersects <u>NO</u> node of body B at same level.

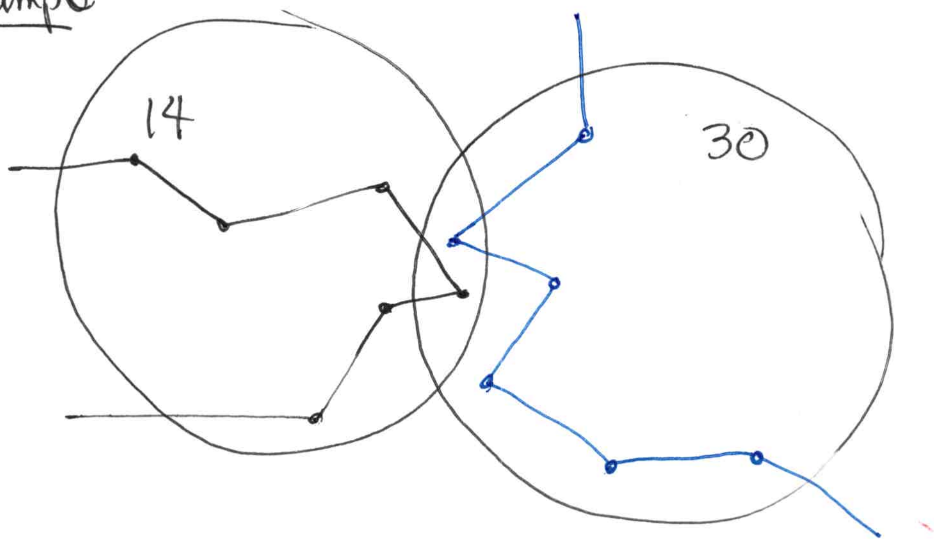
<u>Note:</u> Bounding volume hierarchy rule. Union of volumes of body at level i is subset of union at level $i-1$.

2/21/18

7

After all pruning is done, check every remaining leaf of body A against
" " " " " B.

Example



Collision checking Example

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

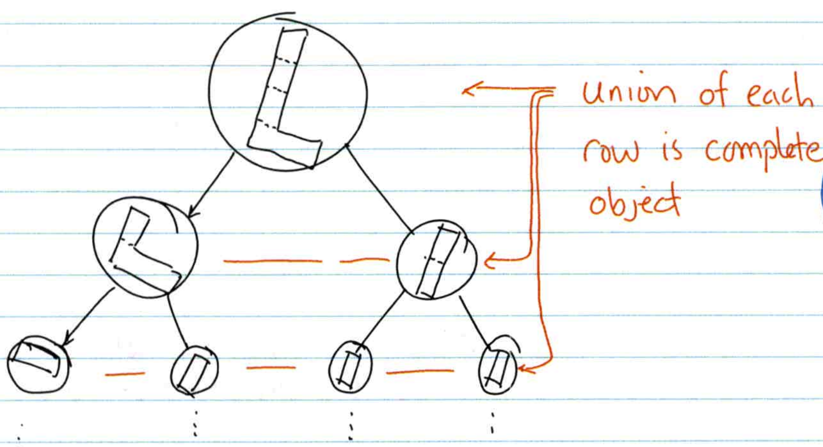
Use hierarchical tree representation of objects.

- 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

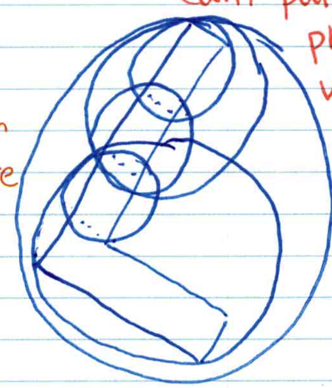
fastest

indep. of details of object geom

Example using spheres (in 2D):



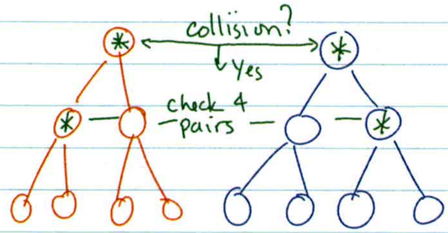
Be careful.
Can't put lowest
piece in
valid
sphere.



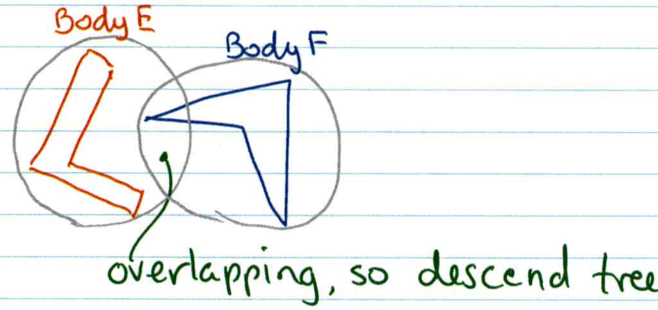
You could keep going until each leaf contains one convex polygon
but you might stop with n triangles or geometric

entities per leaf

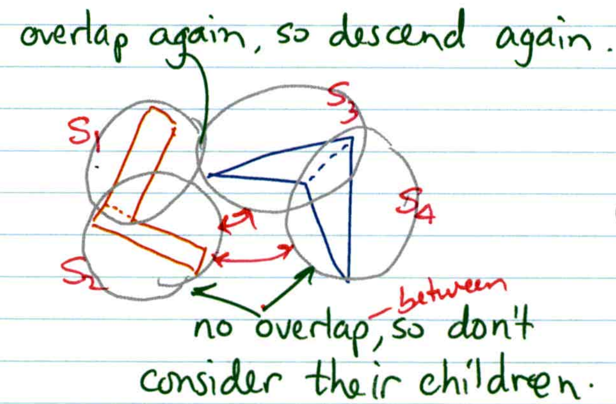
Now consider collision checking with two bodies (two trees):



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



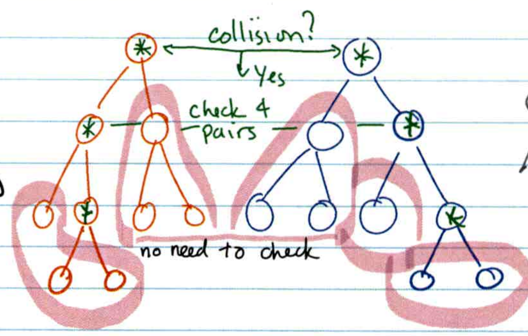
2. If E & F roots intersect, then descend the trees one level

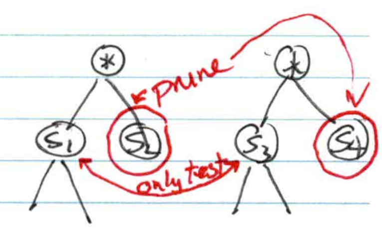


3. Continue recursively until no collisions occur at one level OR we reach the leaves.

4. At the leaves, collision check with geometric primitives

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





leaves is exponential in the depth of the tree
∴ it is important to prune branches

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?

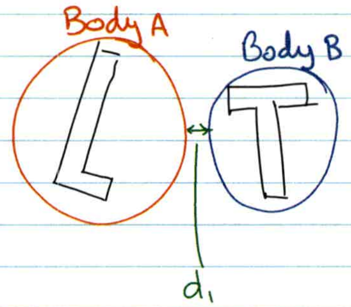
Dynamics! We need ψ_n^l if $\psi_n^l <$ tolerance including $\psi_n^l < 0$ to form timestepping subproblems.

Show video from Dan Negrut's group.

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

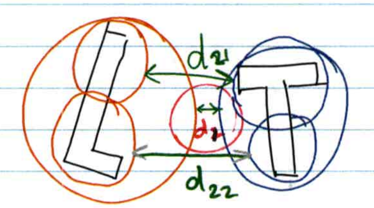
1. Compute distance between root bounding geometries first

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

$d_2 = \min(d_{21}, d_{22}, d_{23}, d_{24})$

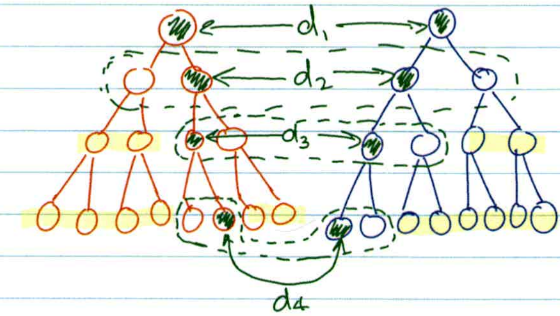
Can we prune ...

$d_{21} \geq d_1$ $d_{23} \geq d_1$

$d_{22} \geq d_1$ $d_{24} \geq d_1$

pairs of subsets → d_{12}

Can we prune like this ?



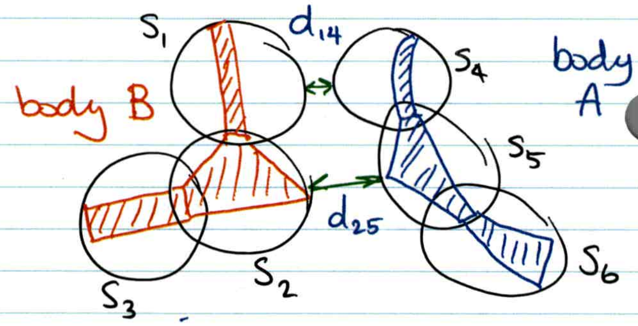
Meaning, can we always prune ~~pairs~~ parts that aren't closest?

Yes. ~~No~~. Prune anything where $d > \bar{d}$ ($\bar{d}$ is dist threshold)

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

Actual distance is d_{25}

$d_{25} > d_{14}$, so can't prune S_2, S_3, S_5, S_6 just because $d_{25} > d_4$ and $d_{36} > d_{14}$.



(if you're only interested in minimum dist)

Possible condition for pruning with bounding spheres?

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

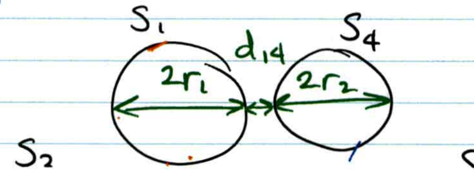
Let d_{ij} be distance between S_i & S_j i on A, j on B

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

You can prune S_k & S_l if

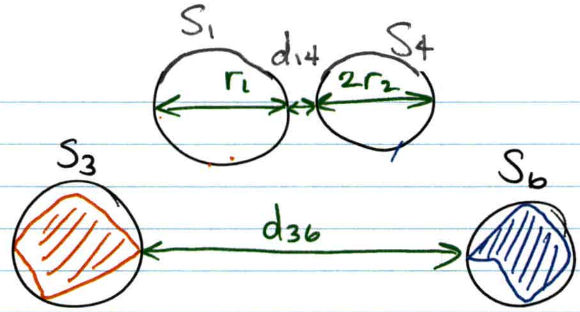
$d_{kl} > d^*_{ij} + 2r_i + 2r_j$

k,l is a pair



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

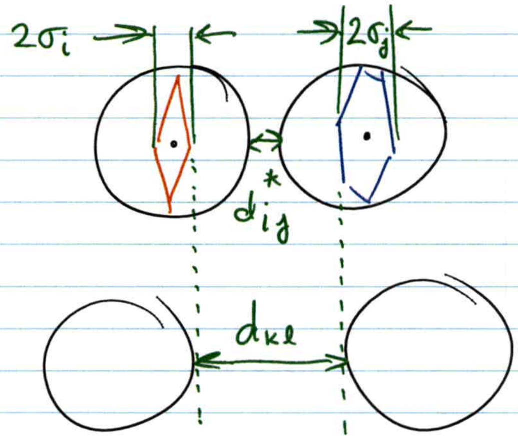
$$d_{36} > d_{14} + 2(r_1 + r_2)$$



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.

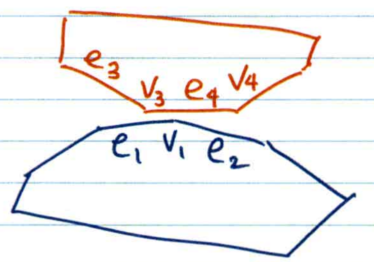
Open question: How can one most efficiently obtain all feature pairs within a given distance (and their distances (including penetration depths))?

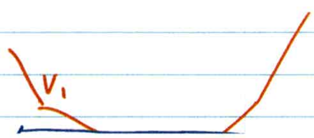
In multibody dynamics, we need to know:

$\text{dist}(e_1, v_3) < \epsilon$

$\text{dist}(v_1, e_4) < \epsilon$

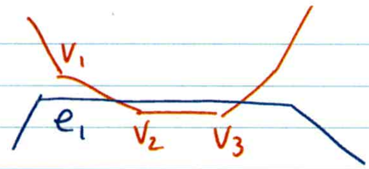
$\text{dist}(v_4, e_2) < \epsilon$





$dist(v_4, e_2) < \epsilon$

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

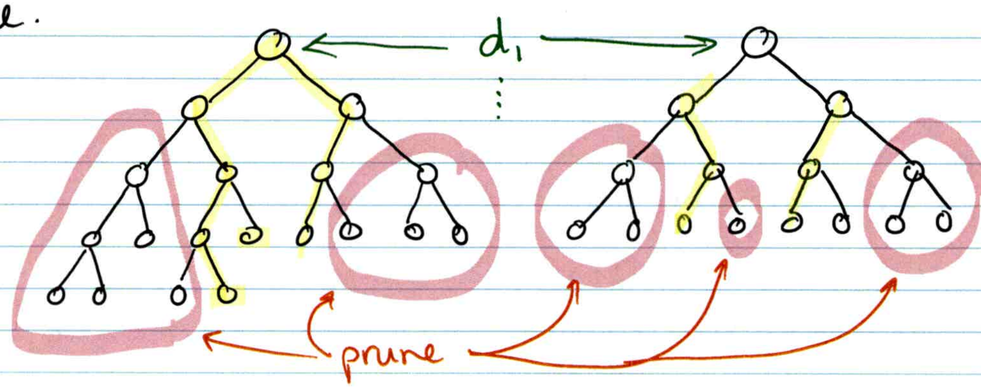


here we need $(v_1, e_1), (v_2, e_1), (v_3, e_1)$

Modify Pruning condition

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

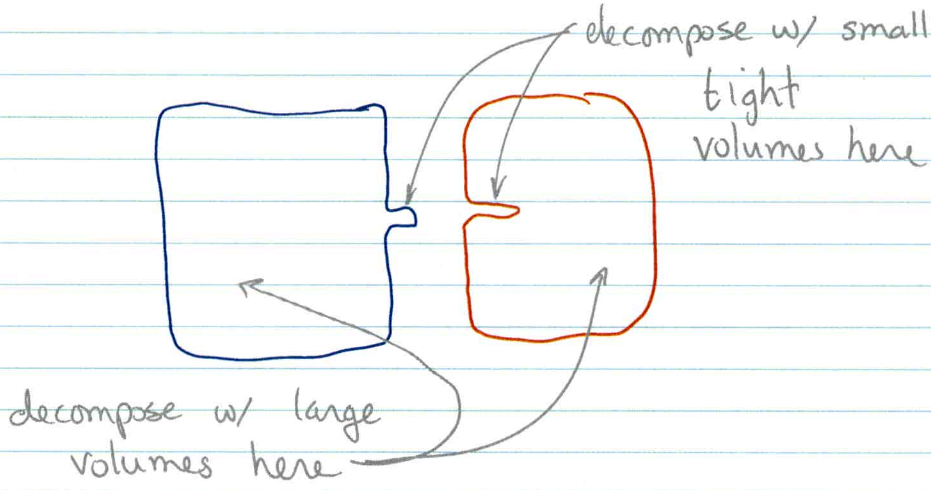
then prune.



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

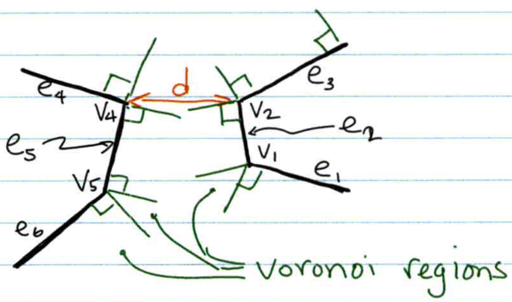
Use spatial coherence to speed collision checking.

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.

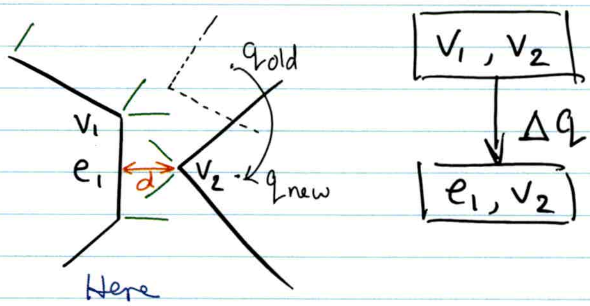
Polygon distance condition

Let F_1 & F_2 be geometric features of convex 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 likely to contain the closest points



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

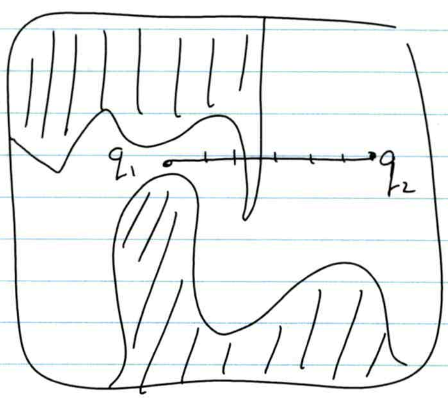
---

Collision checking along path segments

Sample path checking for collision at each point.

Could use multi-resolution scheme. e.g. Van der Corput

If collision found, then discard path segment



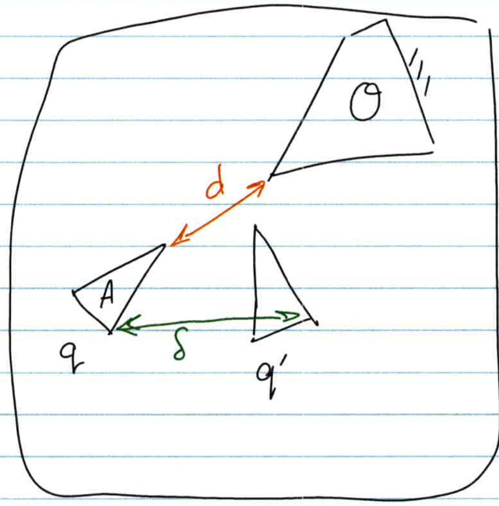
How can we guarantee that a continuous path segment is 100% collision free?

Derive bounds. Unfortunately they tend to be loose.

Planar Case

Let d be the distance at q

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



$q, q' \in SE(3)$

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

However if δ denotes the greatest displacement of 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 $(x_t, y_t) \in \mathbb{R}^2$, $\theta \in S'$

Translation (θ fixed)

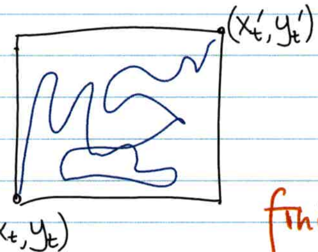
If A translates from (x_t, y_t) to (x_t', y_t'), then

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

~~(Assumes that travel is directly from q to q')~~

Suppose we only require path to stay in a box

Then max δ of $a \in A$ is bounded by



$$\delta \le \cancel{\sqrt{(x_t - x_t')^2 + (y_t - y_t')^2}}\ \underbrace{|x_t - x_t'| + |y_t - y_t'|} \quad \forall a \in A$$

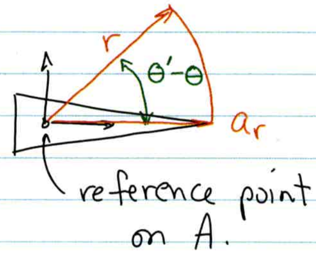
This is a looser approx, but it avoids sqrt(). This is the 1-norm.

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

the shortest path is S'.

This is the 1-norm. Manhattan dist.

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



Assume Rotation Only

The greatest displacement any point $a \in A$ can experience is bounded by $r|\theta' - \theta|$ ← rotation is direct from θ to θ'.

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

can tighten bound by $\sqrt{(x_t - x_t')^2 + (y_t - y_t')^2}$

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

Lavalle's eq (5.3.1)

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

If you know d and r (from previous comp.) then compute l.h.s. changes until l.h.s. = d. You can use bisection search.

An analogous approach can be used in 3D worlds.

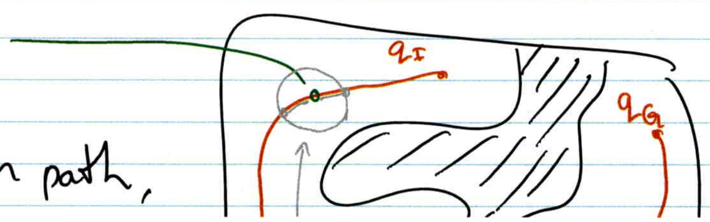
When the robot has multiple links, the approach becomes strongly configuration dependent, making it difficult to apply cost-effectively.

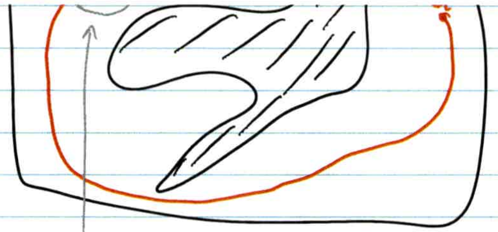
How do we choose the step we can take?

Choose q

Compute d

For each point, q, on path,





For each point, q, on path,

$\Delta x = |x_t - x_t'|$,

$\Delta y = |y_t - y_t'|$, and

$r\Delta\theta = r|\theta_t - \theta_t'|$ are known

Find $q \ni \Delta x + \Delta y + r\Delta\theta < d$

Lenght of path can become very small in "tight" spaces since d becomes very small.