Environment models

Parking Team

23/05/19

How can the environment be modelled?

1. Graph Representation
2. Cell Decomposition
   - Accurate Decomposition
   - Approximate Decomposition
3. Road map
   - Visibility Graph
   - Voronoi Graph
   - Triangulation
4. Potential Field
5. Sampling-Based Path Planning
   - RRT method (single path search)
   - PRM method (multiple path search queries)

First…

Environment =

free space + occupied space

↓

start and goal configurations

(= set of parameters defining robot in space)

1. Graph Representation

free space =

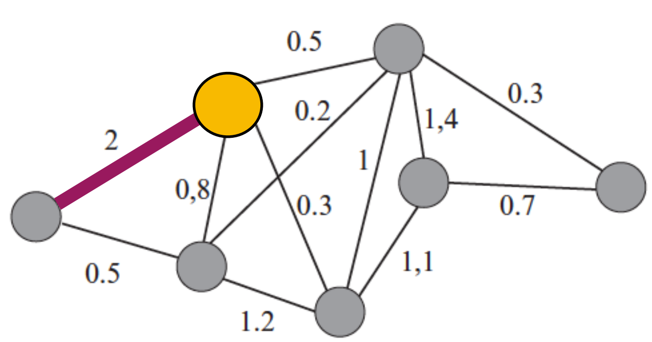
all possible configurations of mobile system

subset of configurations:

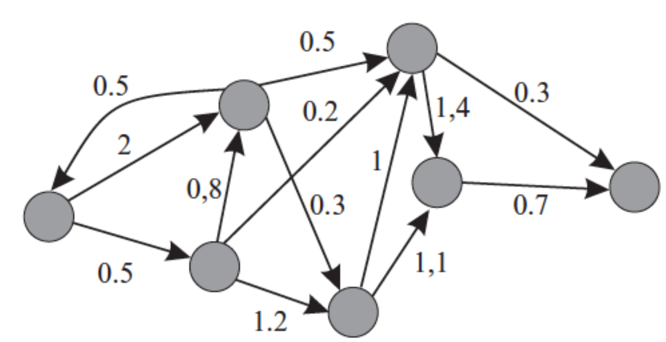
- start/goal
- intermediate
- transitions

start/goal, intermediate	state	nodes
transitions	transition	connections
	graph	

weighted graph

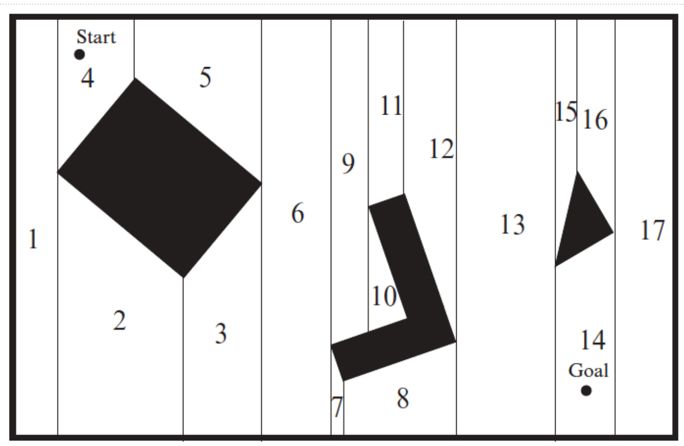
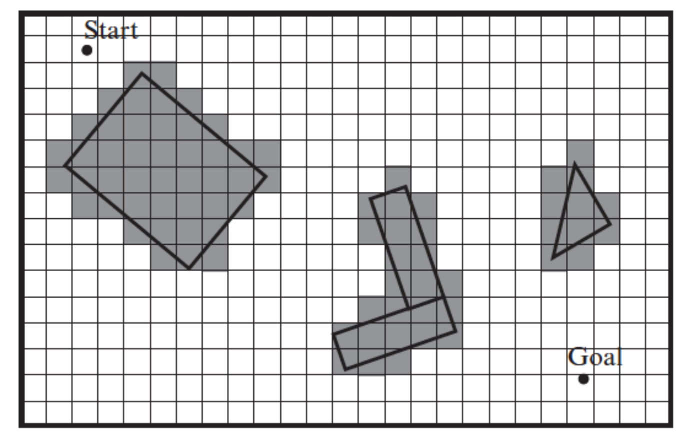
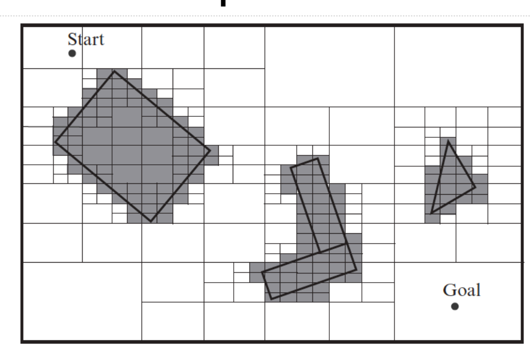


directed graph



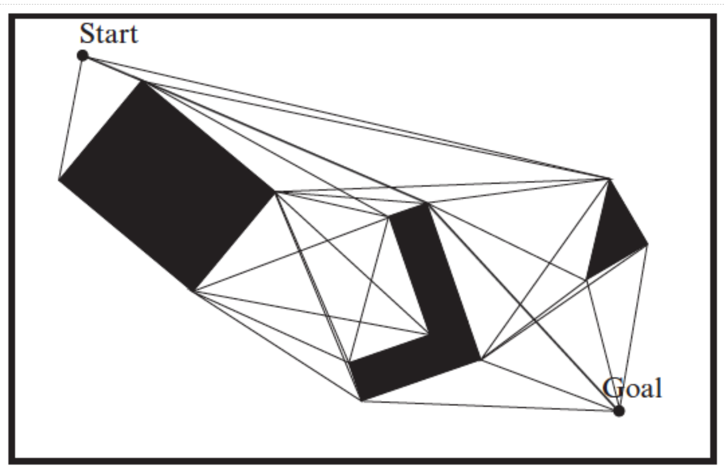
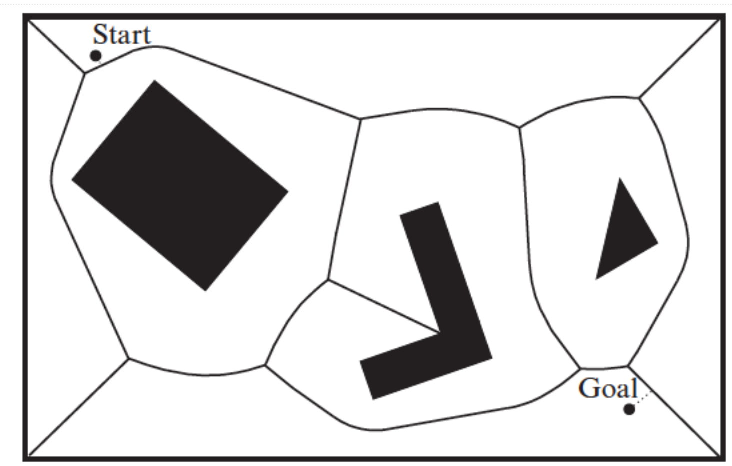
2. Cell Decomposition

environment partitioned to cells

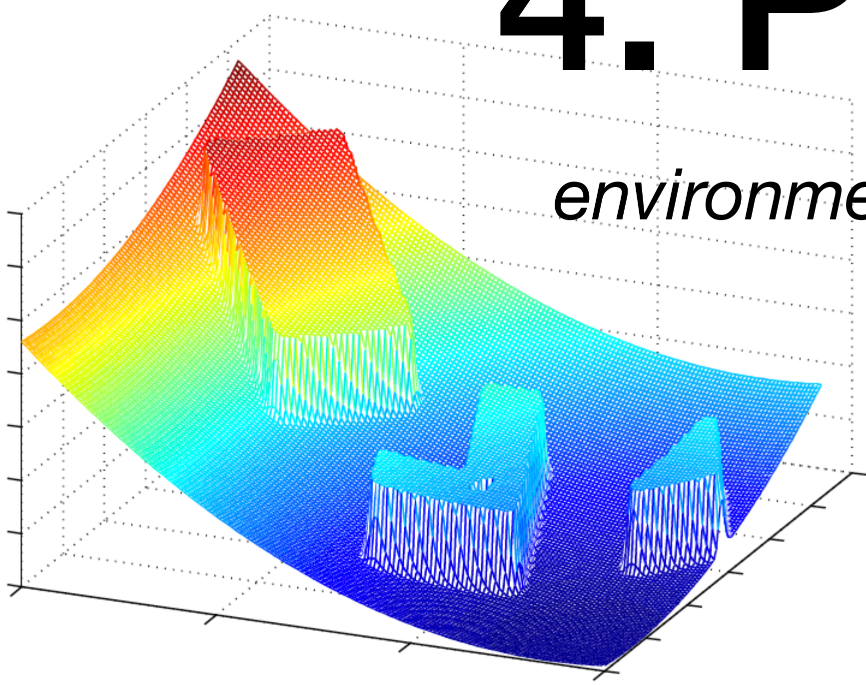
Accurate Decomposition	**Approximate Decomposition**
all cells entirely in free space or entirely in occupied space	same cells contain free space and (part of) an obstacle
	
+ no losses — number of obstacles increases —> number of cells increases	+ simpler — losses — memory usage fix: variable cell size —> quadtree:
	

3. Roadmaps

give possible connections between points

Visibility Graph	**Voronoi Graph**
roads = all possible connections among - points in free space - neighbouring points of same obstacle	roads= borders of regions, generated by obstacles: largest distance from obstacle 3 Voronoi curves
	
result = path as close as possible to obstacles (—) —> risk of collision shortest path possible (+)	result = path maximising distance to obstacle —> minimising risk of collision (+) path unnecessarily long (—)
suitable for: robots using only touch/vicinity sensors	suitable for: robots using distance sensors

4. Potential Field



environment as potential field: imaginary height

potential field =

attractive field + repulsive field

$$U_{\text{attr}}(\boldsymbol{q}) = k_{\text{attr}} \frac{1}{2} D^2(\boldsymbol{q}, \boldsymbol{q}_{\text{goal}}) \qquad U_{\text{rep}}(\boldsymbol{q}) = \begin{cases} \frac{1}{2} k_{\text{rep}} \left(\frac{1}{D(\boldsymbol{q}, \boldsymbol{q}_{\text{obst}})} - \frac{1}{D_0} \right)^2; & D(\boldsymbol{q}) \leq D_0 \\ 0; & D(\boldsymbol{q}) > D_0 \end{cases}$$

path obtained by following negative gradient of potential field

“ball rolling downhill”

— problem: robot may be trapped in local minimum

—> if obstacles have concave shape

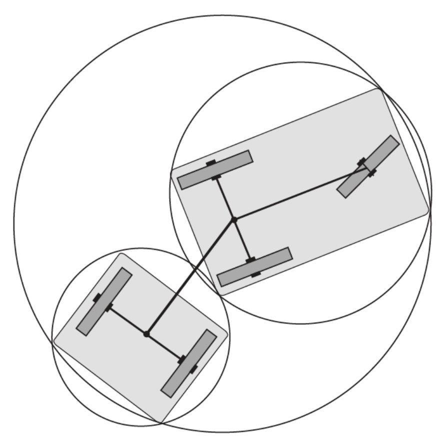
—> if robot oscillated between equally distanced points to obstacle

5. Sampling-Based Path Planning

free space randomly sampled and dynamically expanded

sampling-based methods:

random points sampled —> collision detection to determine if point belongs to free space



simplified by:

- replacing shapes by circles
- exchanging bigger shapes by smaller shapes

motivation:

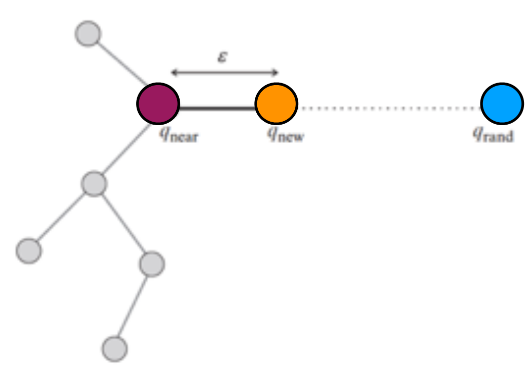
- in general: no time-consuming calculation of free space
- compared to Decomposition: no high number of cells
- compared to Potential Fields: problem of being trapped unlikely

sampling-based approaches

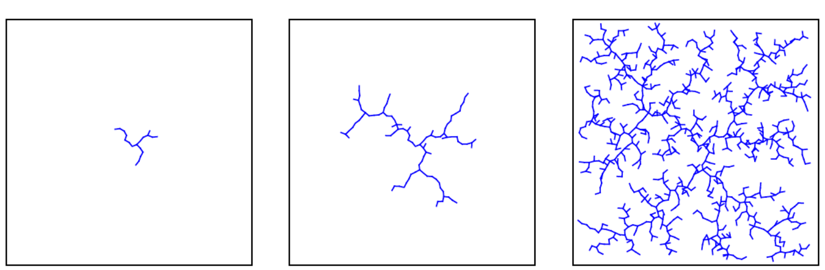
RRT Method
(single path search)

each iteration:
new connection added to graph
1st iteration:
graph = starting configuration
next iteration:

- choose random point
- search closest node from existing graph
- calculate new node
- new node in free space? —> add to graph



+ only 2 parameters
+ consistent/ simple



PRM Method
(multiple path search queries)

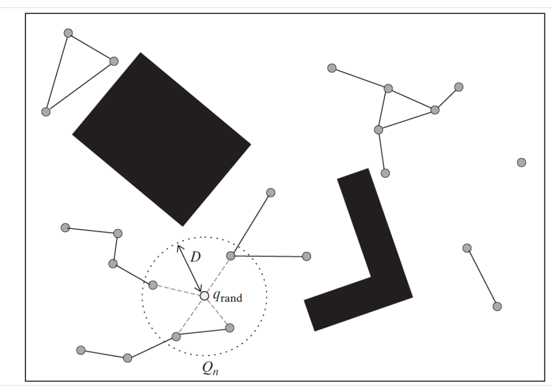
2 steps

1) *learning phase*

- choose random point from free space
- find Qn to expand map
- add connection from random point to Qn that lie in free space

2) *path searching*

- connect start/goal to closest nodes of map
- apply path searching algorithm

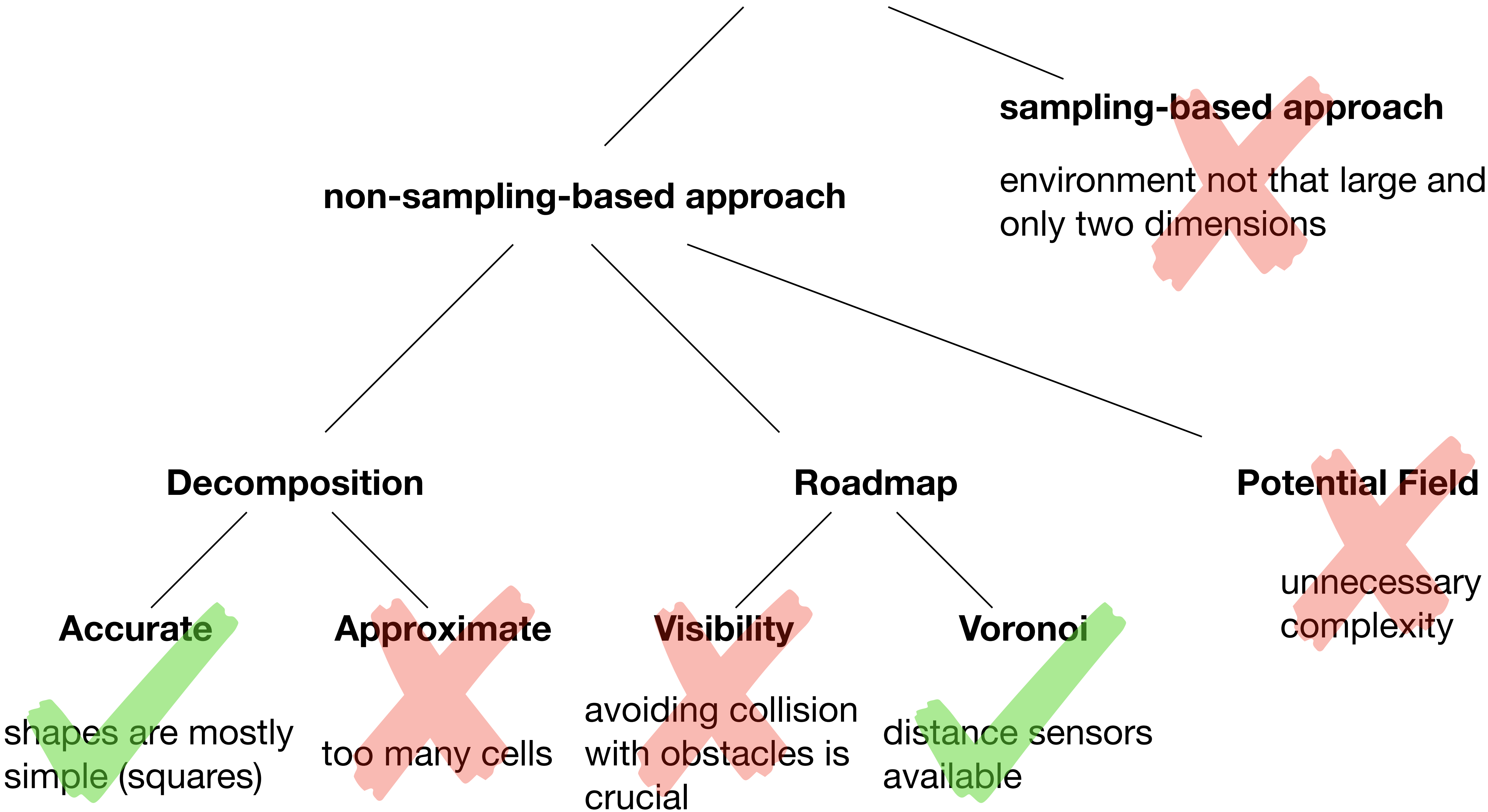


+ effective for robots with many DOF

— finding path through narrow passage

fix: bridge test

Which models are useful for us?



Questions? :)