Motion Planning

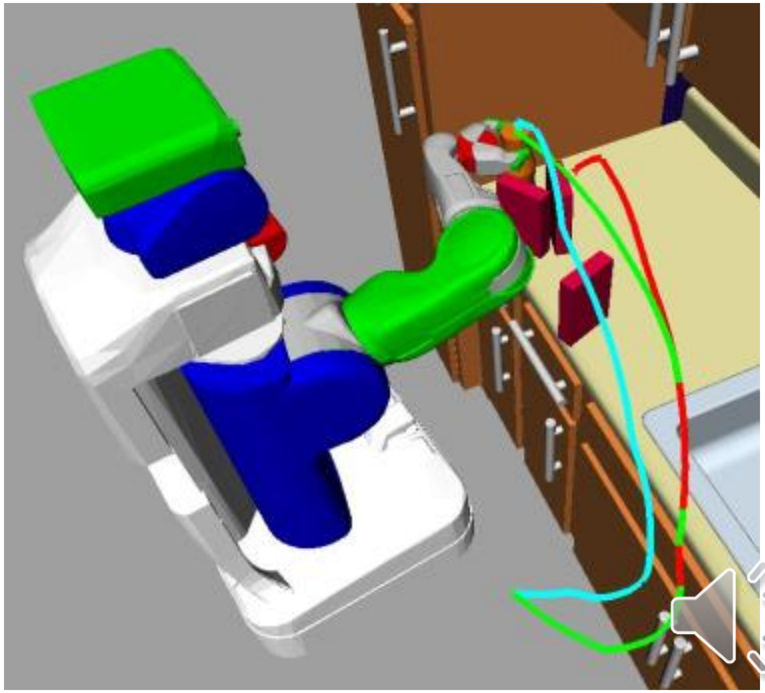
Robotics – COMS4045

Benjamin Rosman

Some material taken from slides by: Pieter Abbeel, Steven LaValle, James Kuffner and Howie Choset

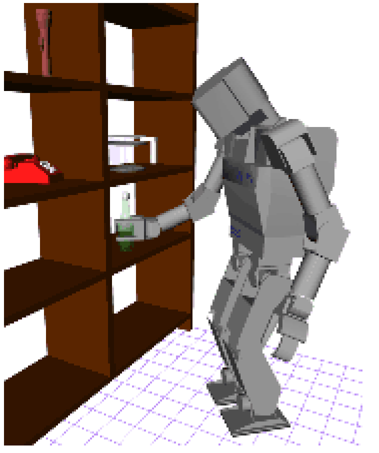
What is Motion Planning?

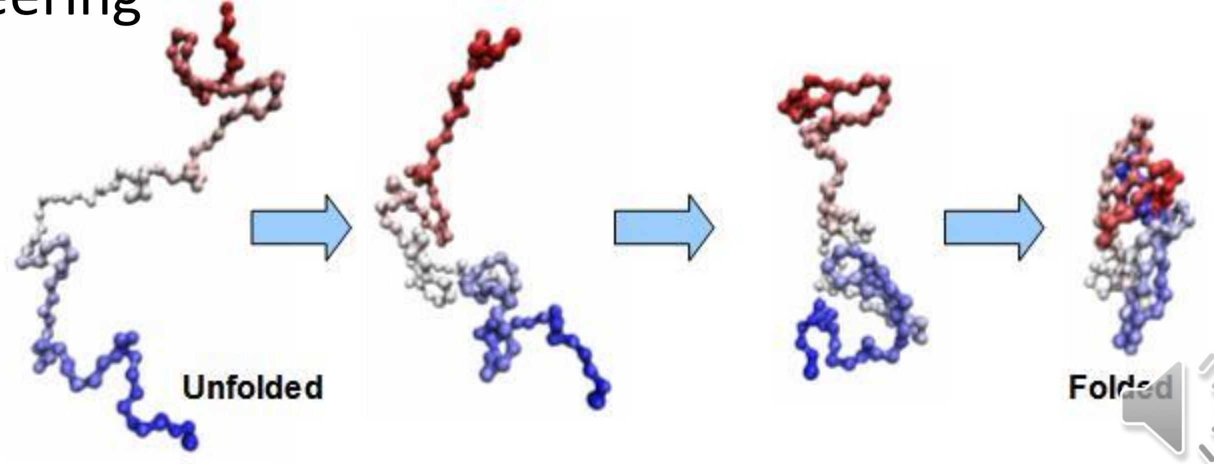
- Given a robot, a source and a destination
  - Find a valid sequence of moves between them
- Geometric search task
  - Often ignores dynamics and differential constraints



What is Motion Planning?

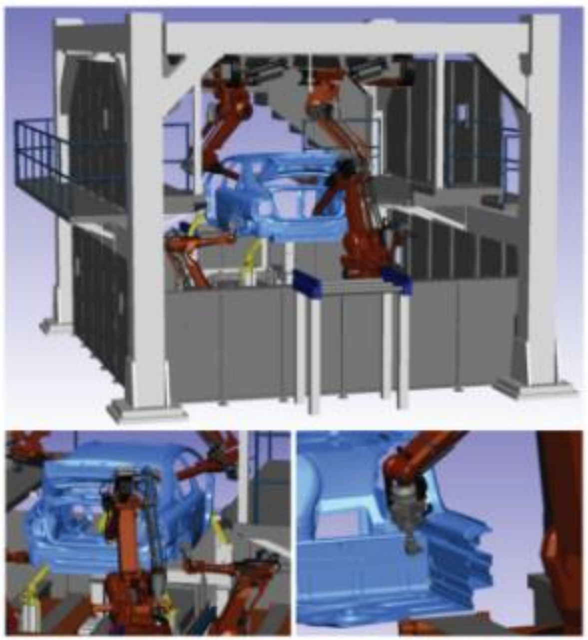
- Difficulties:
  - Continuous state space
  - High dimensional space
  - Dynamic environments
- Other applications:
  - Biology (protein folding)
  - Design engineering
  - Animation

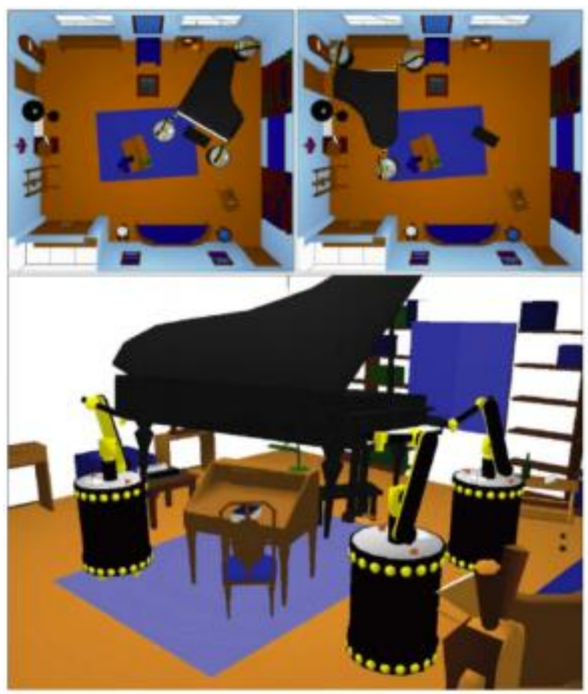


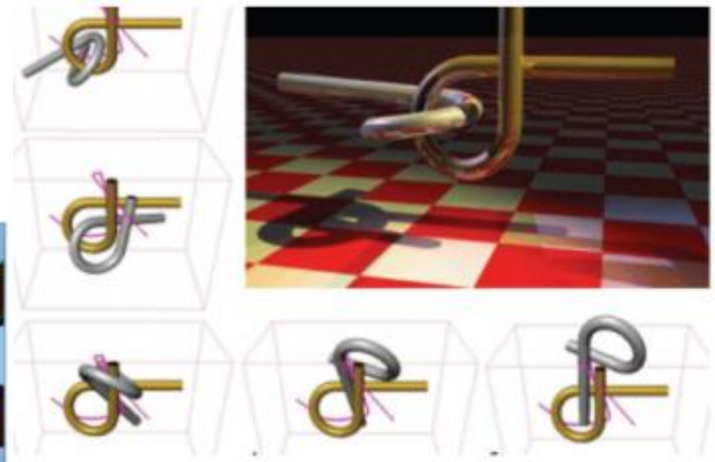


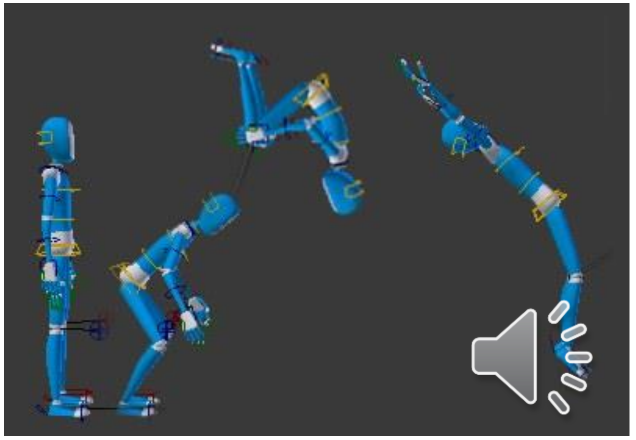
Examples

- How to plan for complicated agents/shapes/environments?



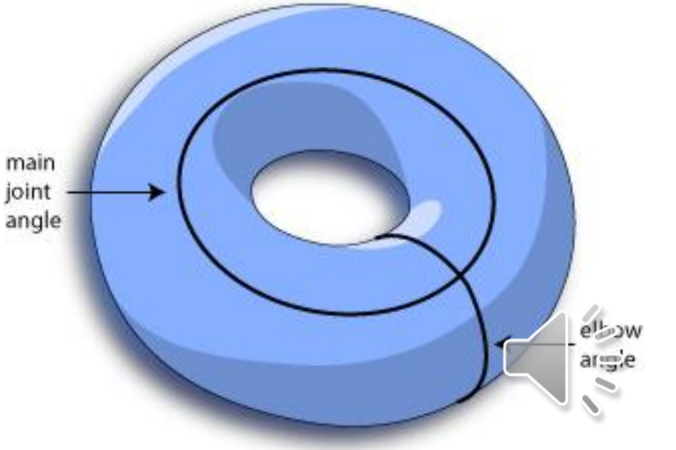




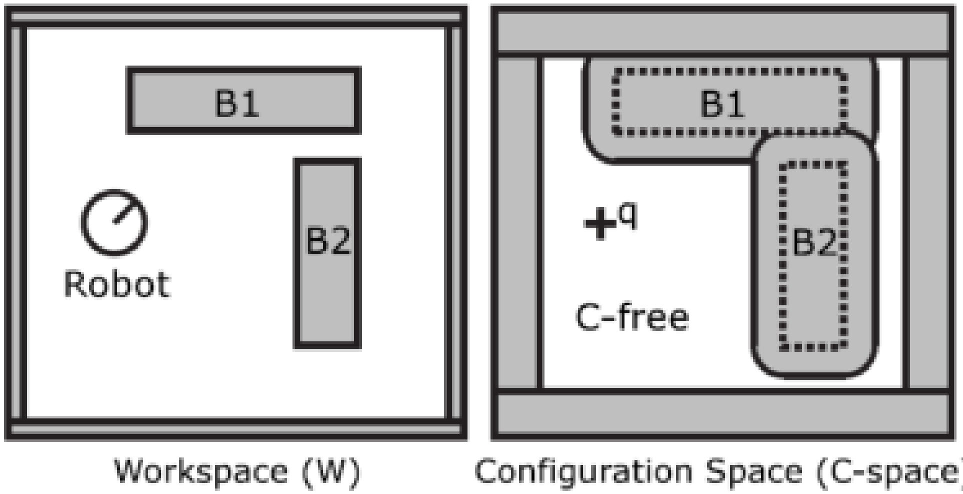


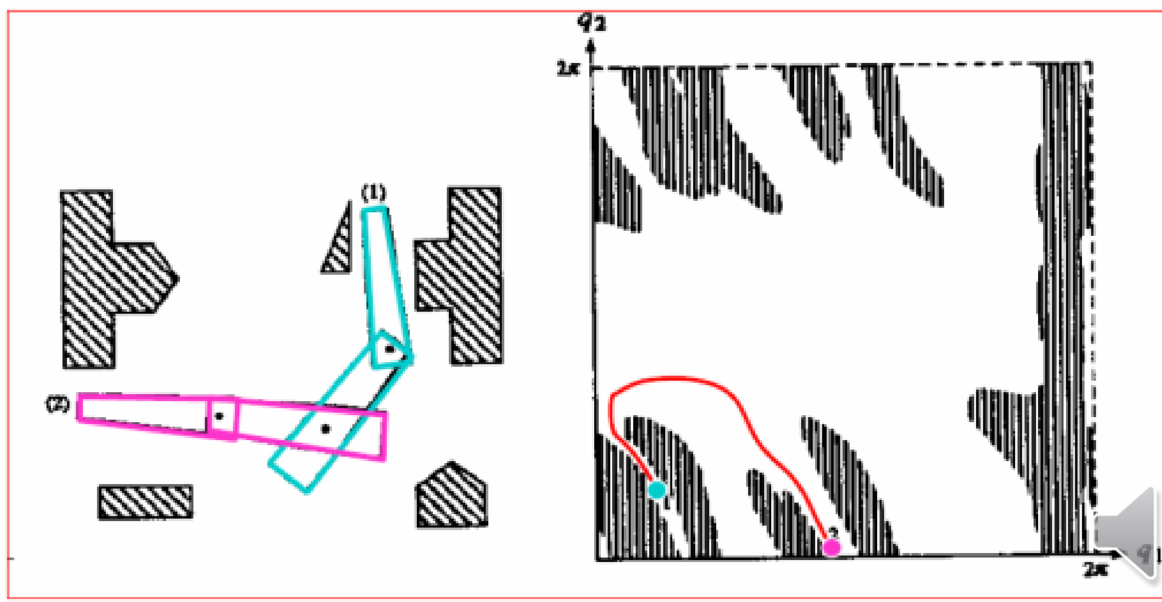
Configuration Space

- C-space = set of all robot configurations q
  – Each dimension corresponds to a degree of freedom
- Free space = set of all allowable/legal configurations
  – Physically reachable by robot
  – No obstacle intersections
- Transformation: work space → configuration space
  – Maps high DoF planning into path planning for point masses in configuration space



C-Space Examples



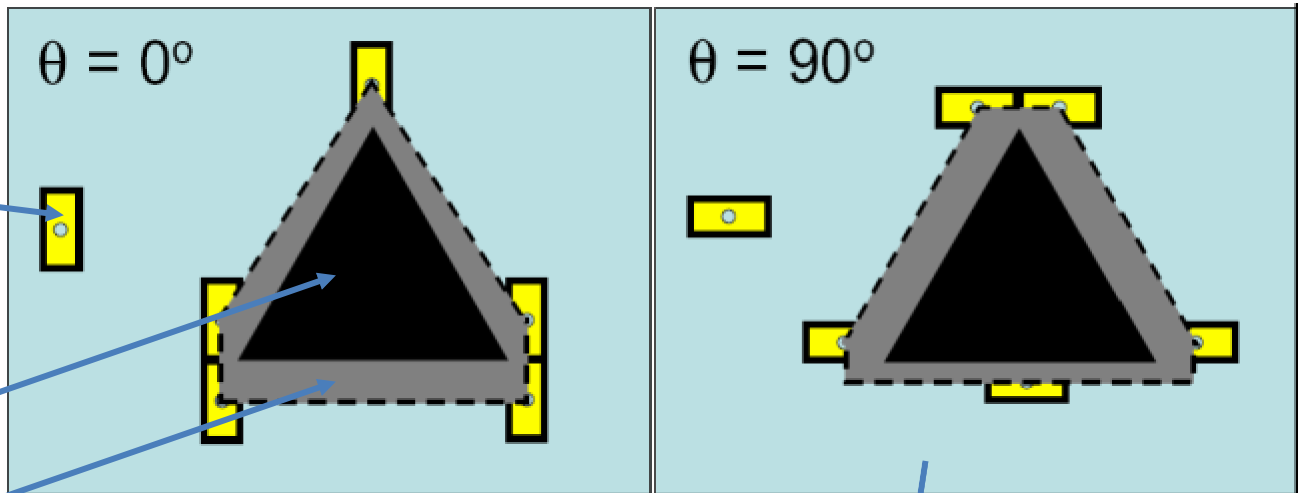


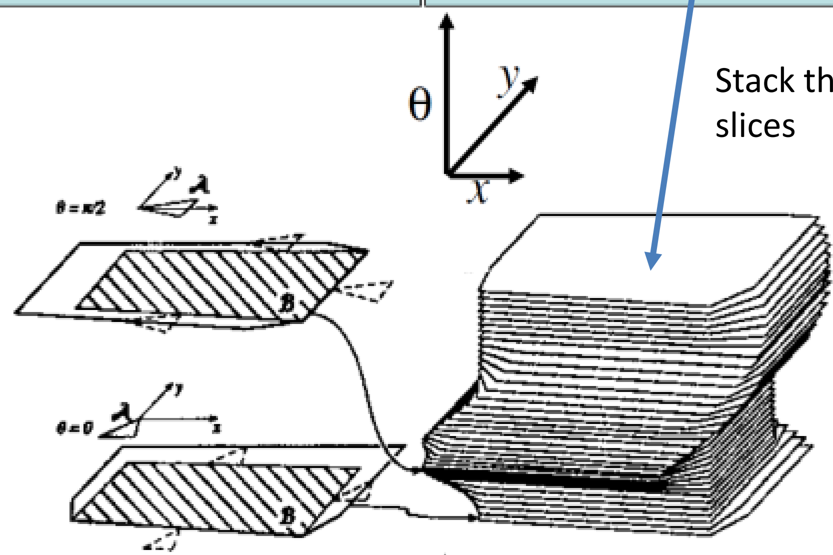
C-Space Dimensionality Changes

Agent:
Can rotate and translate

Triangular obstacle

Non-feasible region in c-space

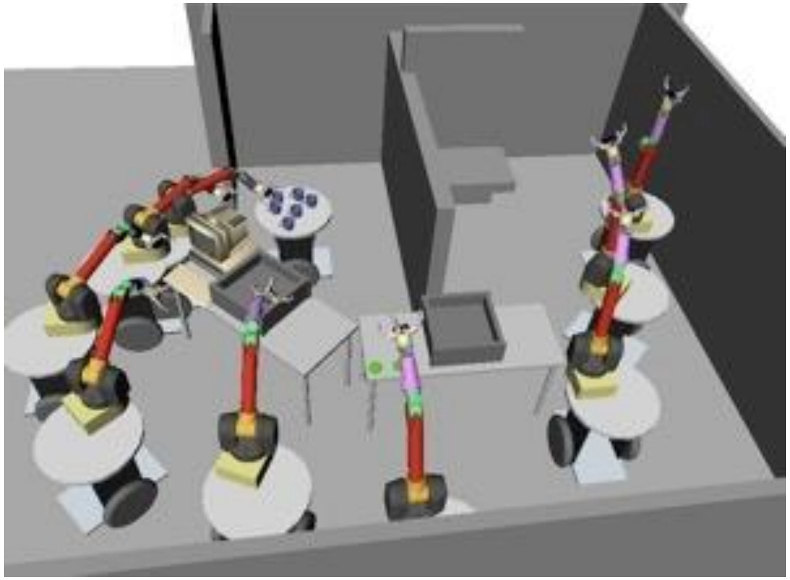




Stack these slices

The Motion Planning Problem

- Input
  - Geometric description of robot and obstacles
  - Initial and goal configurations
- Output
  - Path from initial to goal
  - Or that it doesn't exist
- Key assumption:
  - Map exists!
  - Where from?
  - See lectures on mapping (SLAM)



Approaches

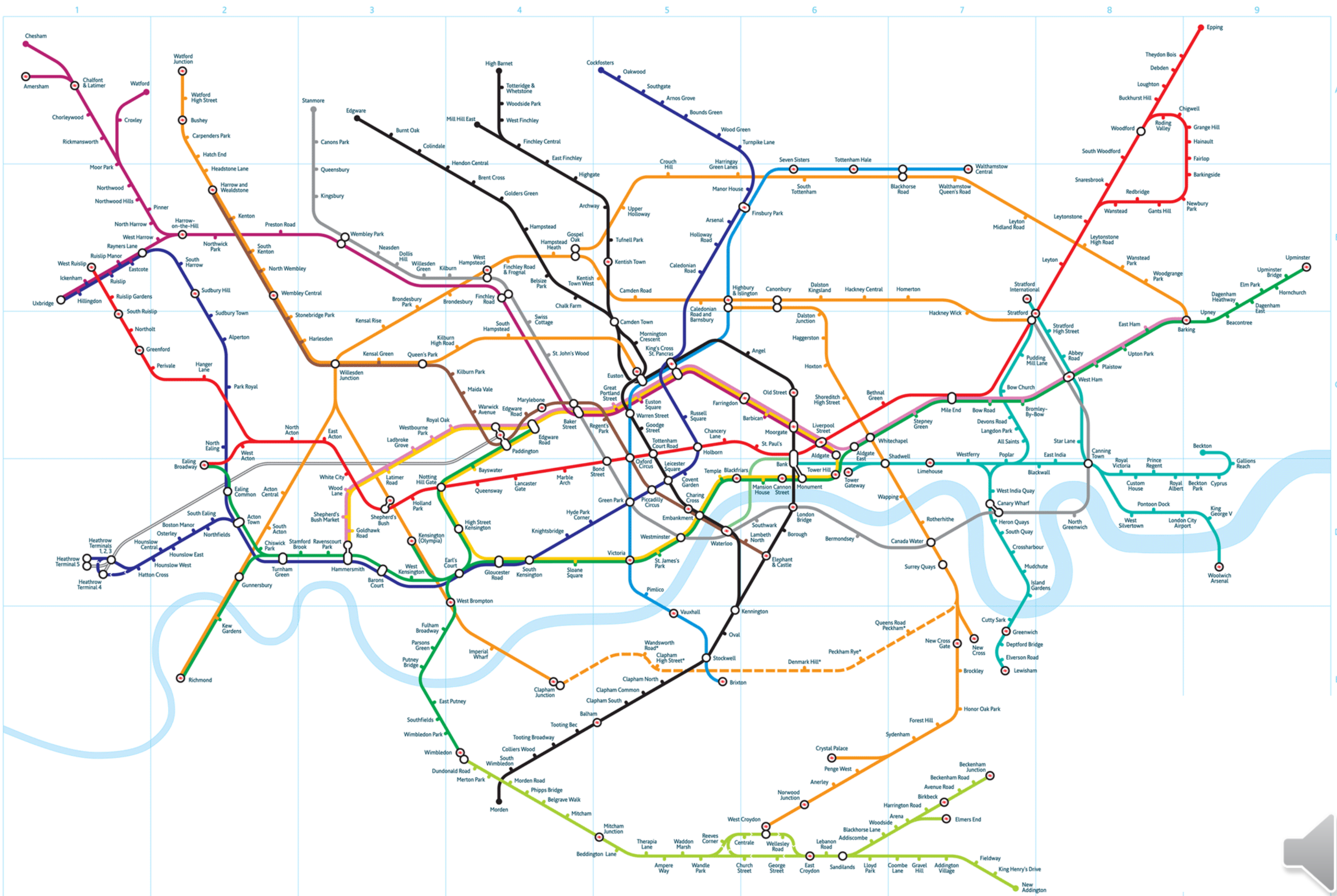
- Often reduce intractable problem in continuous C-space to tractable one in discrete space
- Define functions over the space
  – Nice, closed form solutions
- Approximate the space
  – E.g. road maps (graphs)
- We'll look at several algorithms
  – No single best one, depends on the problem

Roadmaps

- Pre-compute (small) graph over free space
  - Discrete approximation
- Find path from start to goal using roadmap
  - Shortest path planning on graph
  - E.g. A*

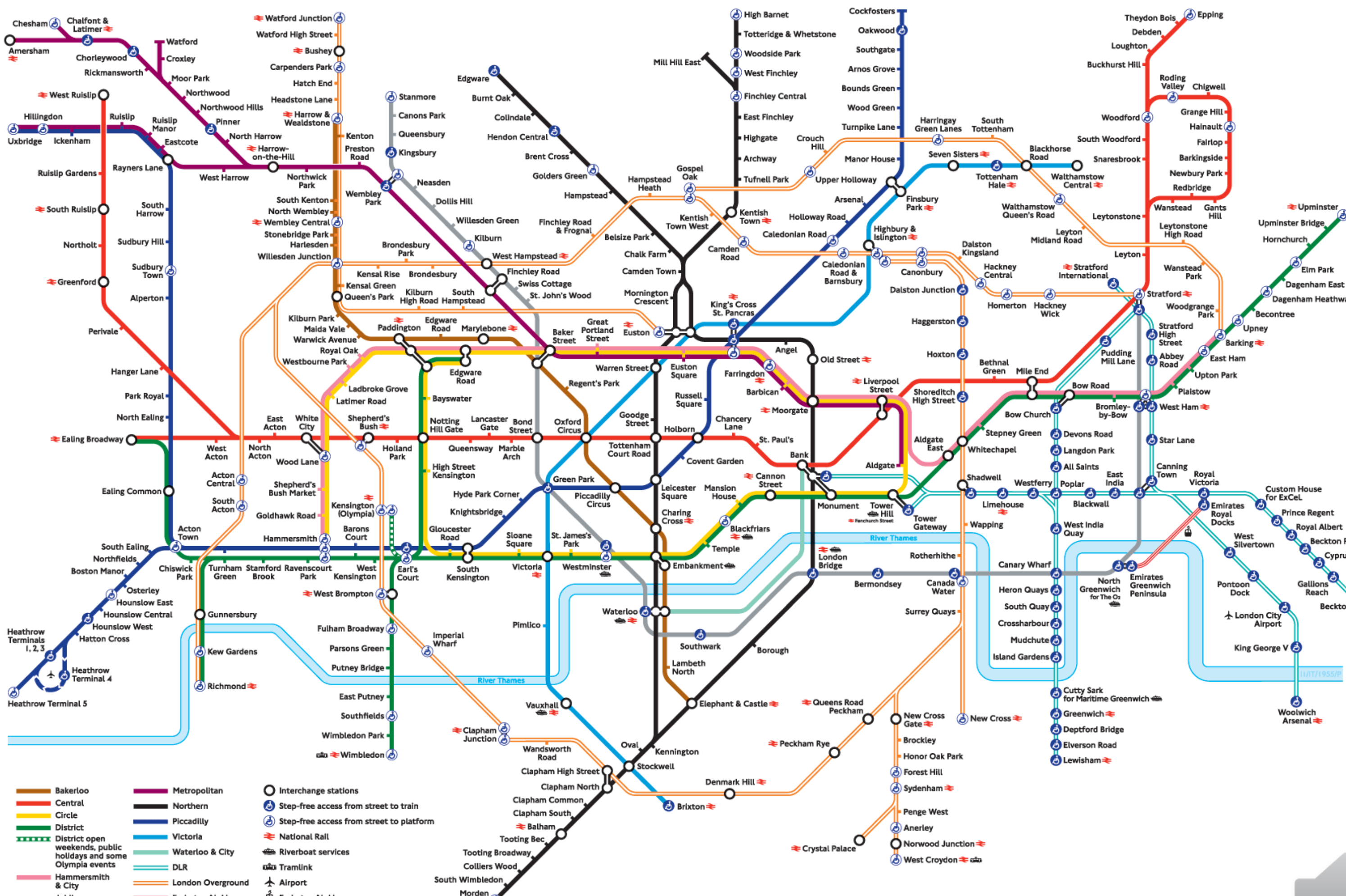
Roadmaps

Roadmaps



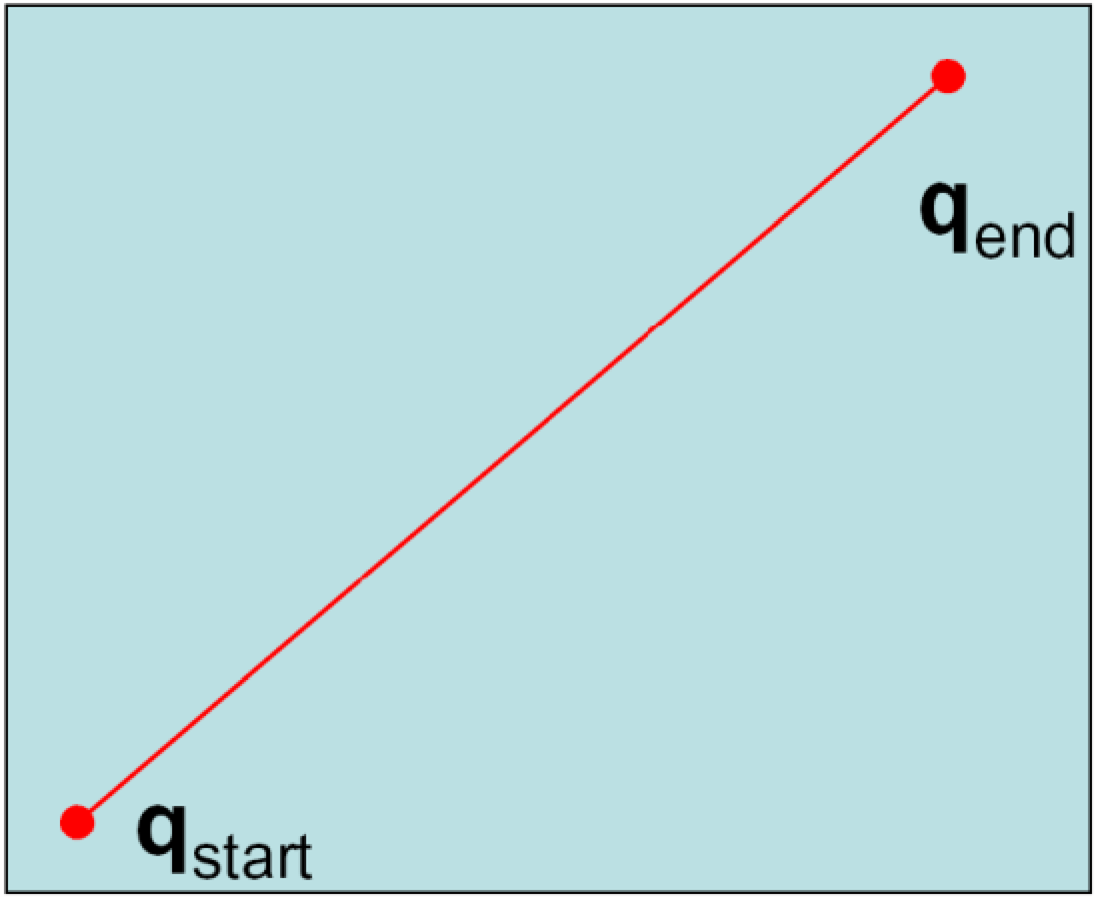
Chesham
Amersham
Watford
Watford Junction
Stanmore
Edgware
High Barnet
Cockfosters
Epping
Uxbridge
Harrow-on-the-Hill
Wembley Park
Walthamstow Central
Upminster
Richmond
Morden
Brixton
Woolwich Arsenal
Lewisham
Beckton
Stratford
Baker Street
King's Cross St. Pancras
Liverpool Street
Bank
Oxford Circus
Waterloo
Victoria
Paddington
Canary Wharf
West Croydon
New Addington
Elmers End
Beckenham Junction
Crystal Palace
Clapham Junction
Heathrow Terminal 4
Ealing Broadway

Roadmaps


Bakerloo
Central
Circle
District
District open weekends, public holidays and some Olympia events
Hammersmith & City
Jubilee
Metropolitan
Northern
Piccadilly
Victoria
Waterloo & City
DLR
London Overground
Emirates Air Line
Interchange stations
Step-free access from street to train
Step-free access from street to platform
National Rail
Riverboat services
Tramlink
Airport
Emirates Air Line
River Thames
Chesham
Amersham
Watford
Watford Junction
Stanmore
Edgware
High Barnet
Mill Hill East
Cockfosters
Epping
Uxbridge
Heathrow Terminals 1, 2, 3
Heathrow Terminal 4
Heathrow Terminal 5
Ealing Broadway
Richmond
Wimbledon
Morden
Brixton
Upminster
Stratford
Woolwich Arsenal
Lewisham
West Croydon
Crystal Palace
Elephant & Castle
Walthamstow Central
Harrow & Wealdstone
Paddington
King's Cross St. Pancras
Liverpool Street
Bank
Waterloo
Victoria
Canary Wharf

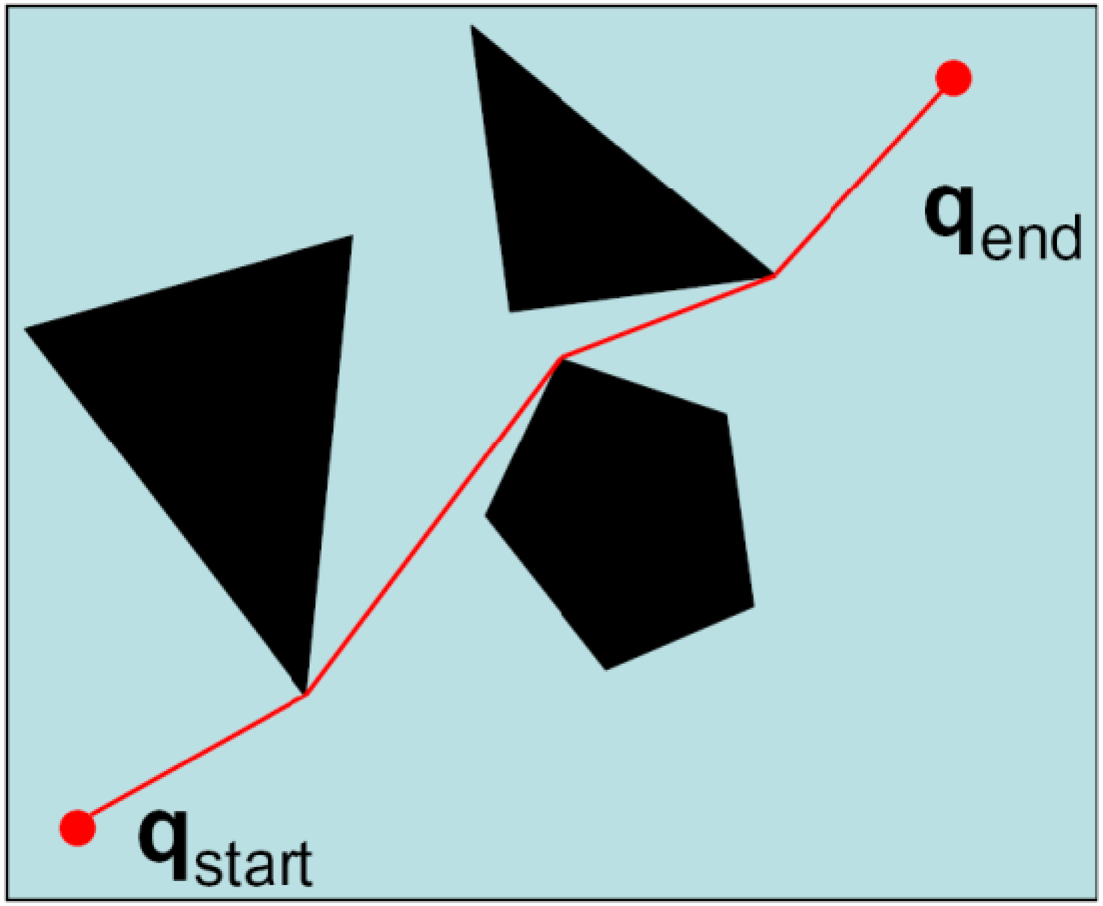
Visibility Graphs

- No obstacles: shortest path is direct line



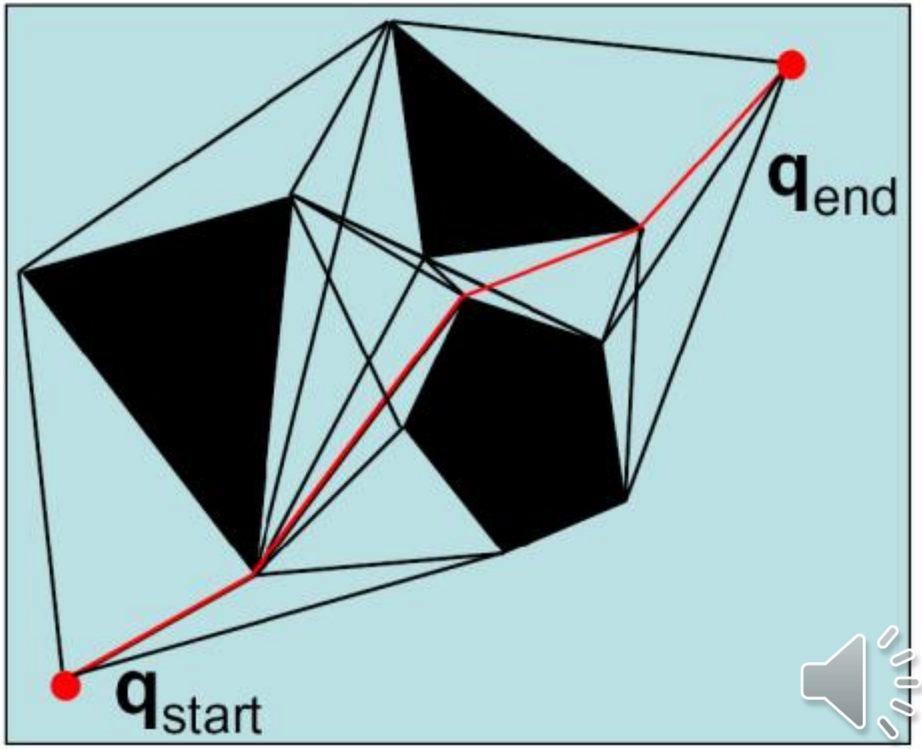
Visibility Graphs

- Polygonal obstacles: sequence of straight lines joining vertices



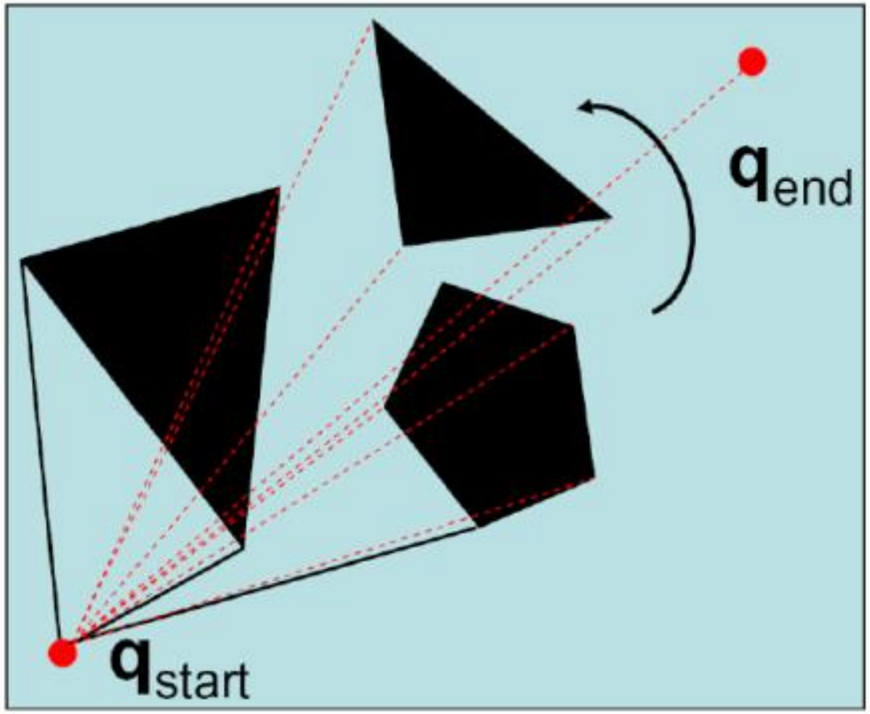
Visibility Graphs

- Visibility graph: set of unblocked lines between obstacle vertices and start/end
  - Every two nodes are linked, if they are visible from each other
  - Solution is shortest path



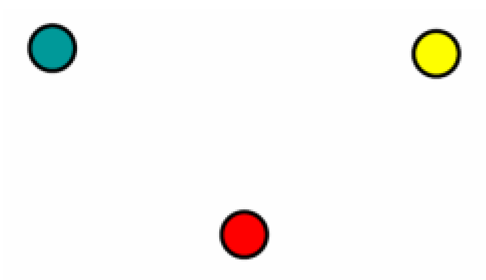
Visibility Graphs

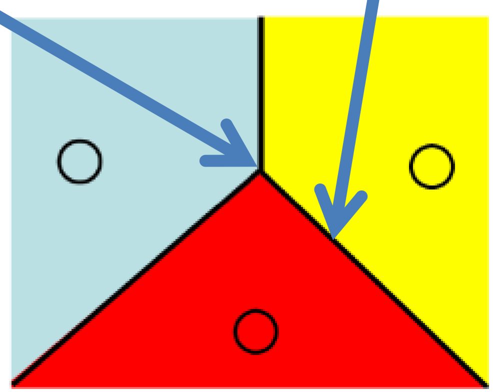
- Construction:
  – Sweep a line from each vertex
  – Record lines that end at visible vertices
- Problems:
  – Stays as close as possible to obstacles – risky (execution errors)
  – Complicated in high dimensions
- Consider optimality/safety trade-off

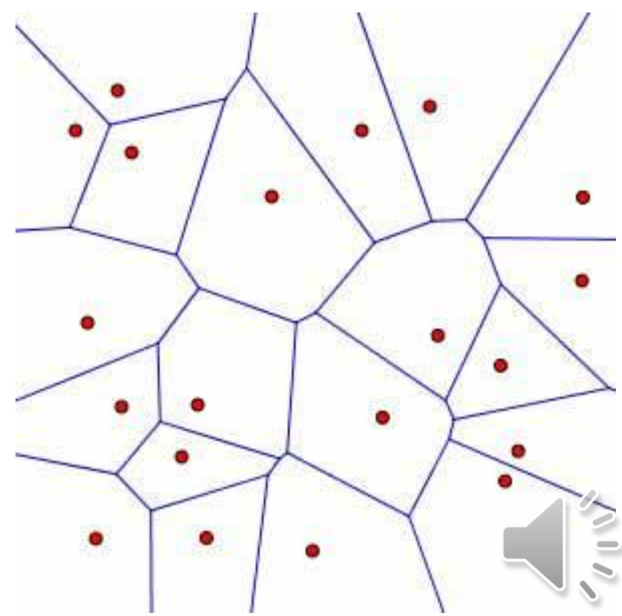


Voronoi Segmentation

- What if we tried to maximise distance from obstacles instead?
- Voronoi diagram
  - Edges are points equidistant from two nodes
  - Vertices are equidistant from three

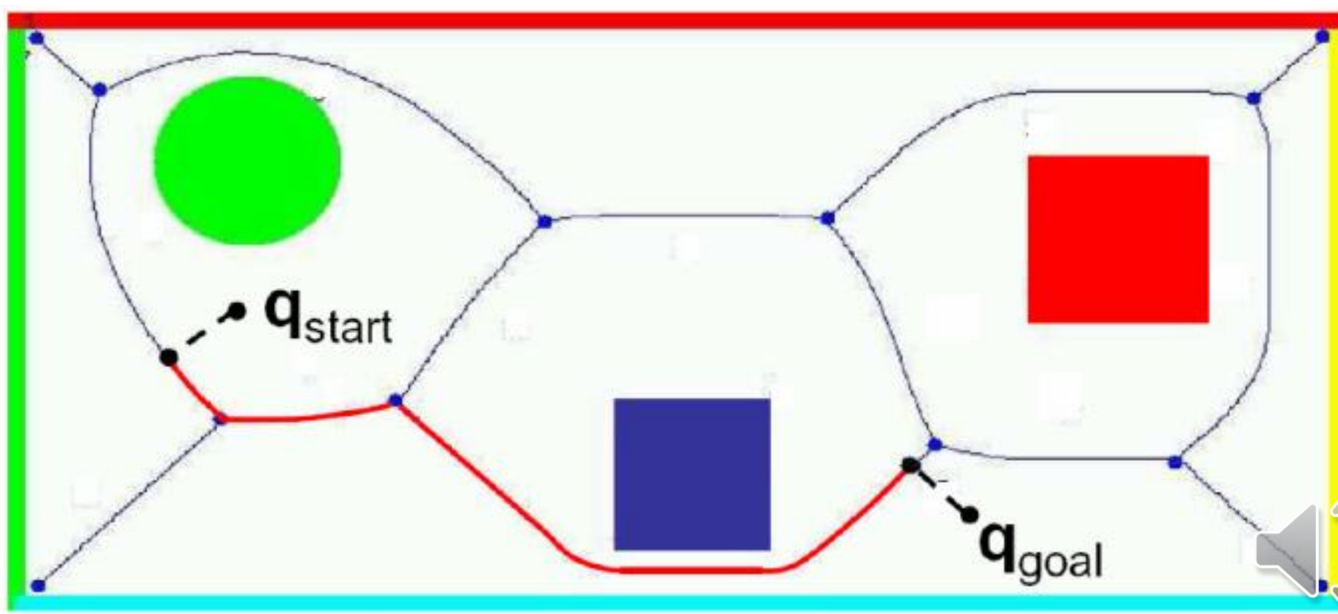






Voronoi Segmentation

- Compute Voronoi segmentation
  - Points on edges are furthest from obstacles
  - Use Voronoi diagram as roadmap
  - Connect start and goal, find closest points on edges
  - Plan along edges

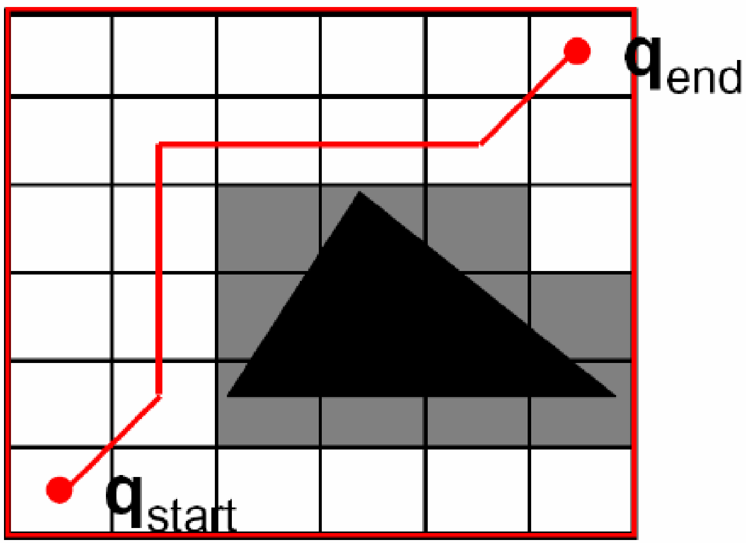


Voronoi Segmentation

- Problems:
  - Difficult with complex (non-polygonal) objects and high dimensions
  - Is this the best heuristic? Very conservative
  - Unstable tessellation (not robust to small changes in obstacle layouts)

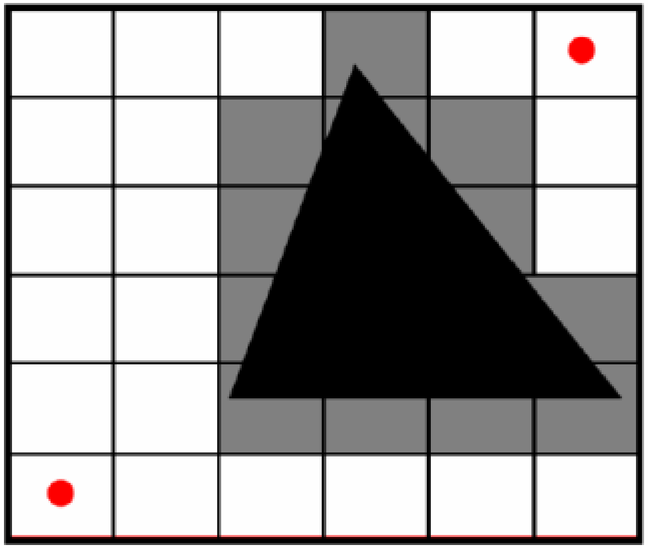
Cell Decomposition – Approximate

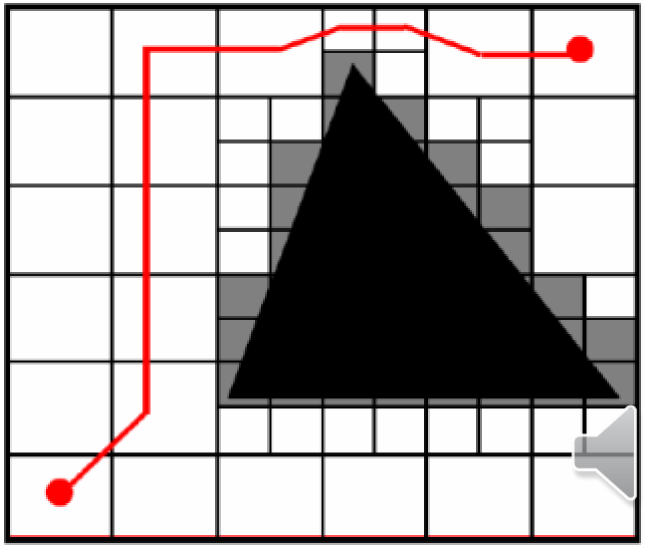
- Define a discrete grid in C-space
  - Mark a cell on the grid intersecting an obstacle as blocked
  - Other cells are free
  - Find path through free space
- Issues:
  - Not complete (miss solutions)
  - Wasting storage in open areas



Cell Decomposition – Approximate

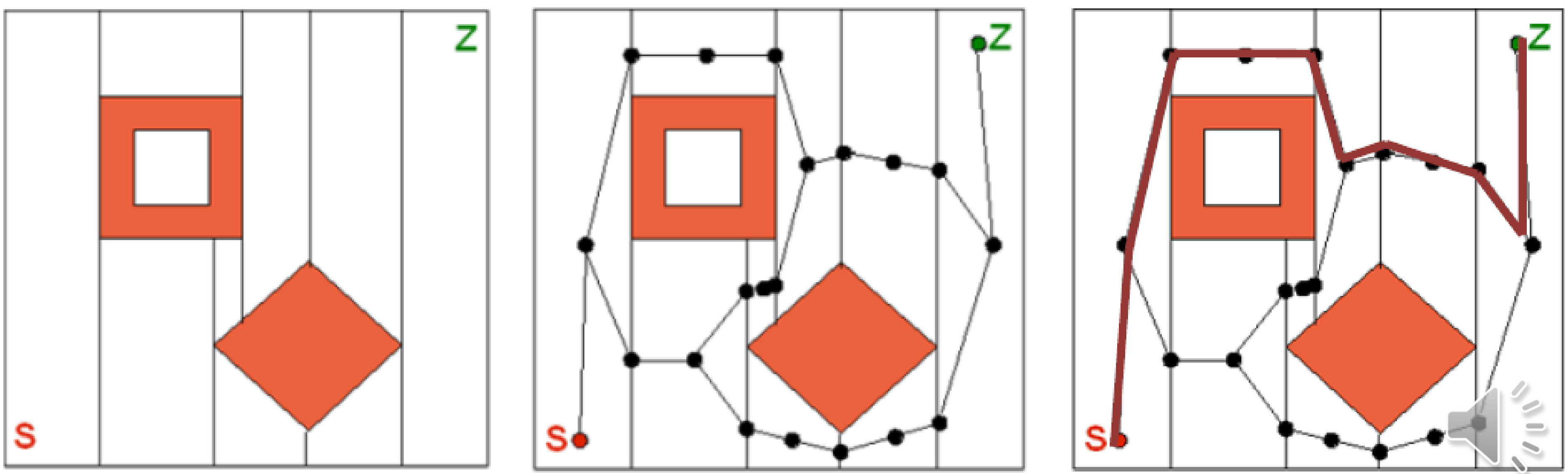
- Issue: not complete
- Solution:
  - Iterative refinement: partially vs fully blocked cells
  - Multi-resolution
- But:
  - High dimensions? Complexity of multiple refinements?



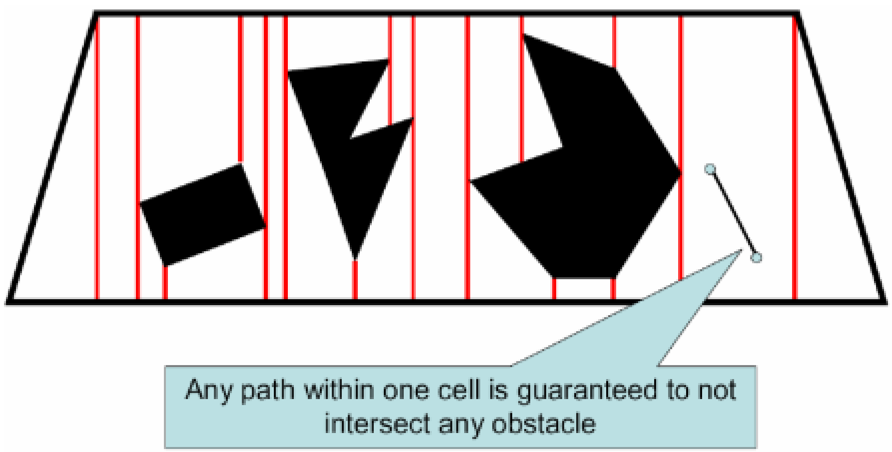


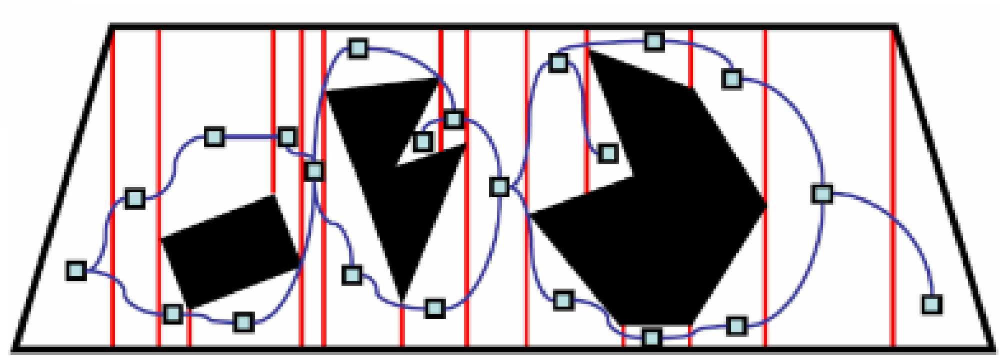
Cell Decomposition – Exact

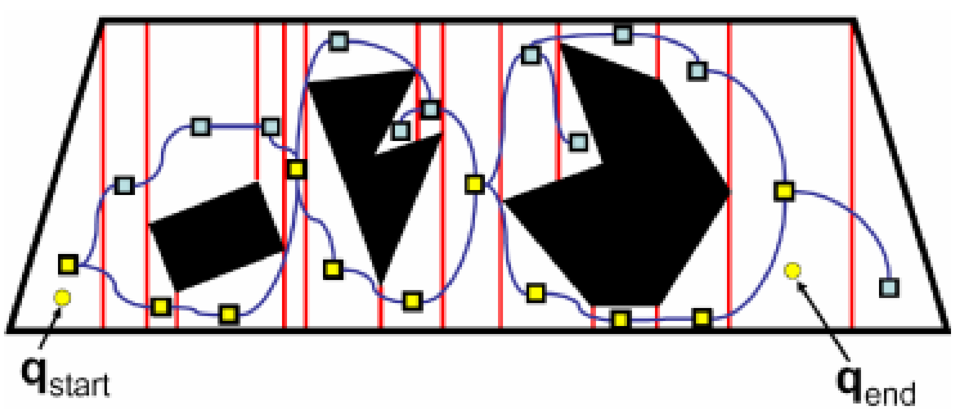
- Decompose free space into trapezoidal cells
- Decompose with vertical lines through vertices
- Represent connectivity by graph adjacency
- Graph is cell centres and edge centres
- Shortest path with graph search



Cell Decomposition – Exact

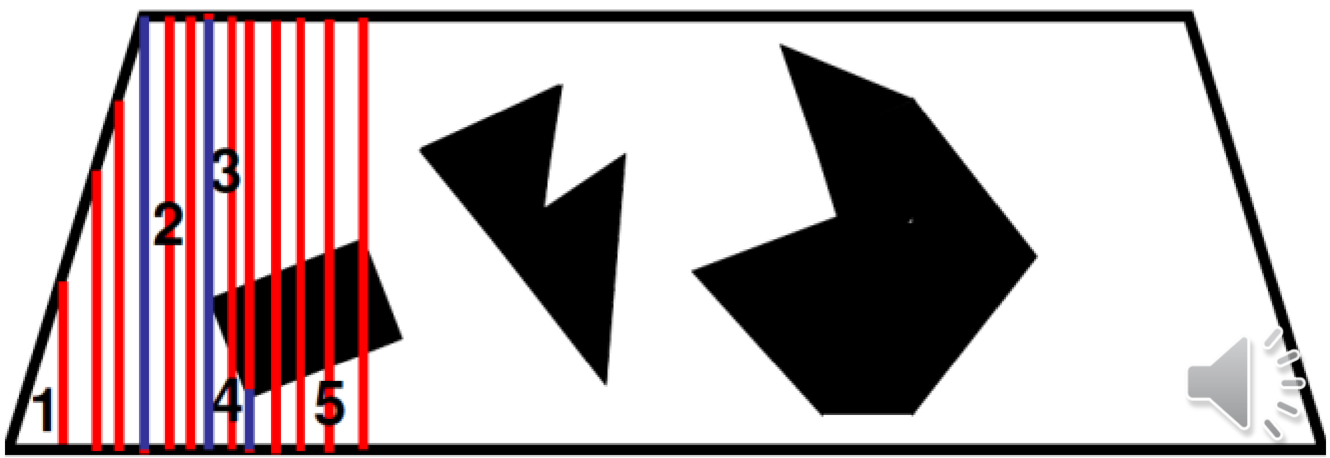






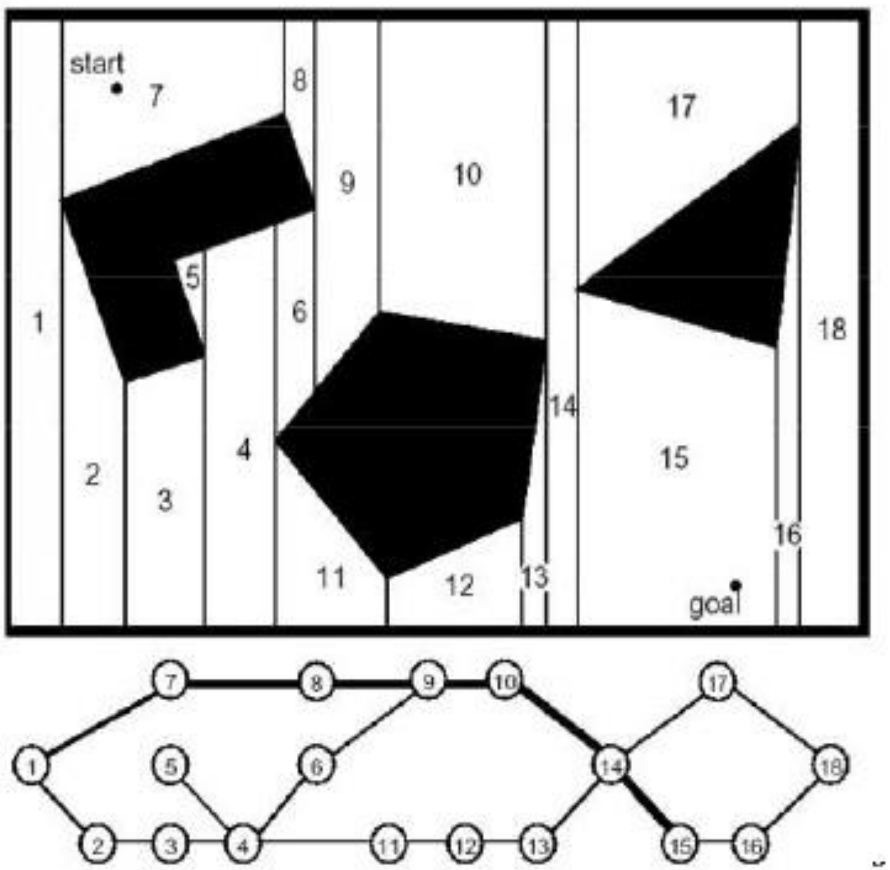
Cell Decomposition – Exact

- Plane sweep algorithm:
  - Order vertices in x direction
  - For each vertex:
    - Construct plane at that x location
    - Check:
      - Split or merge cells
    - Create new cell



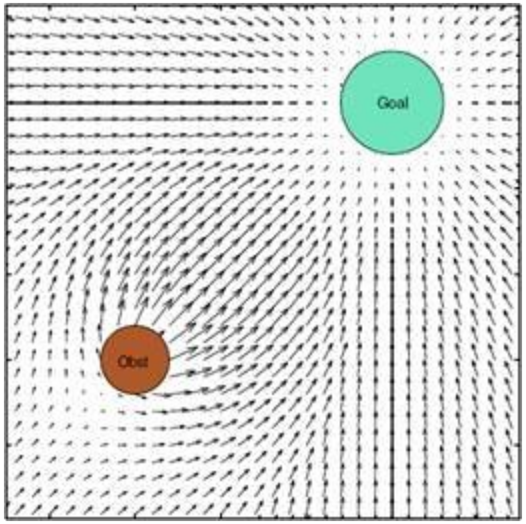
Cell Decomposition – Exact

- Planning graph



Potential Fields

- Define a potential function over free space with global minimum at the goal
- Goal location: attractive potential
- Obstacles: repulsive potential
- Negative gradient of total potential is artificial force on robot
- Follow direction of steepest descent

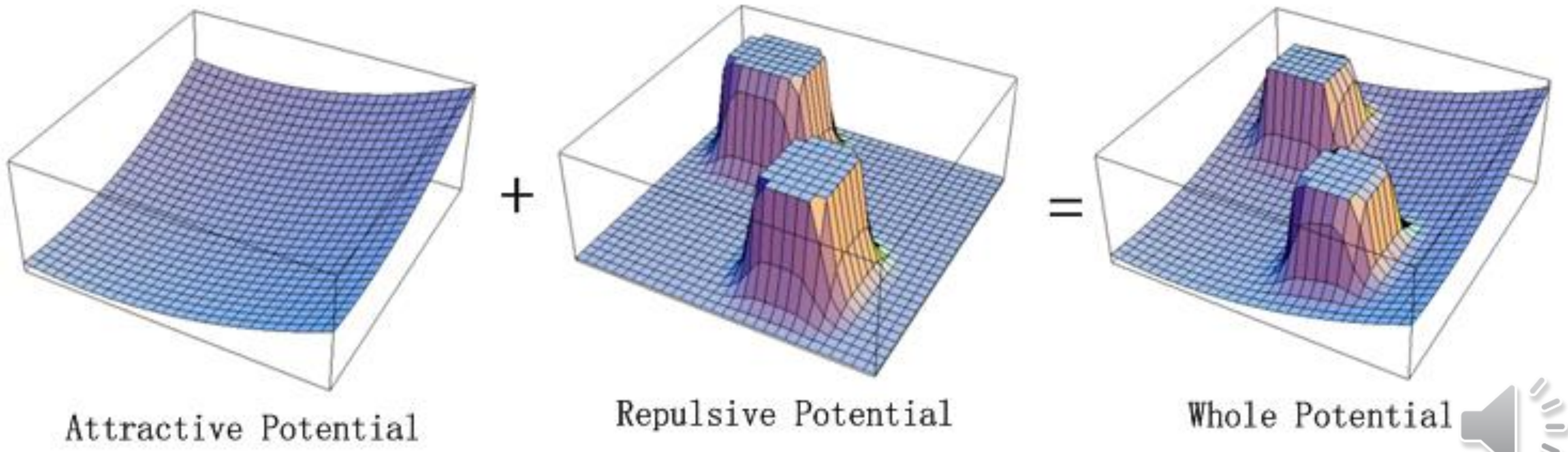


Potential Fields

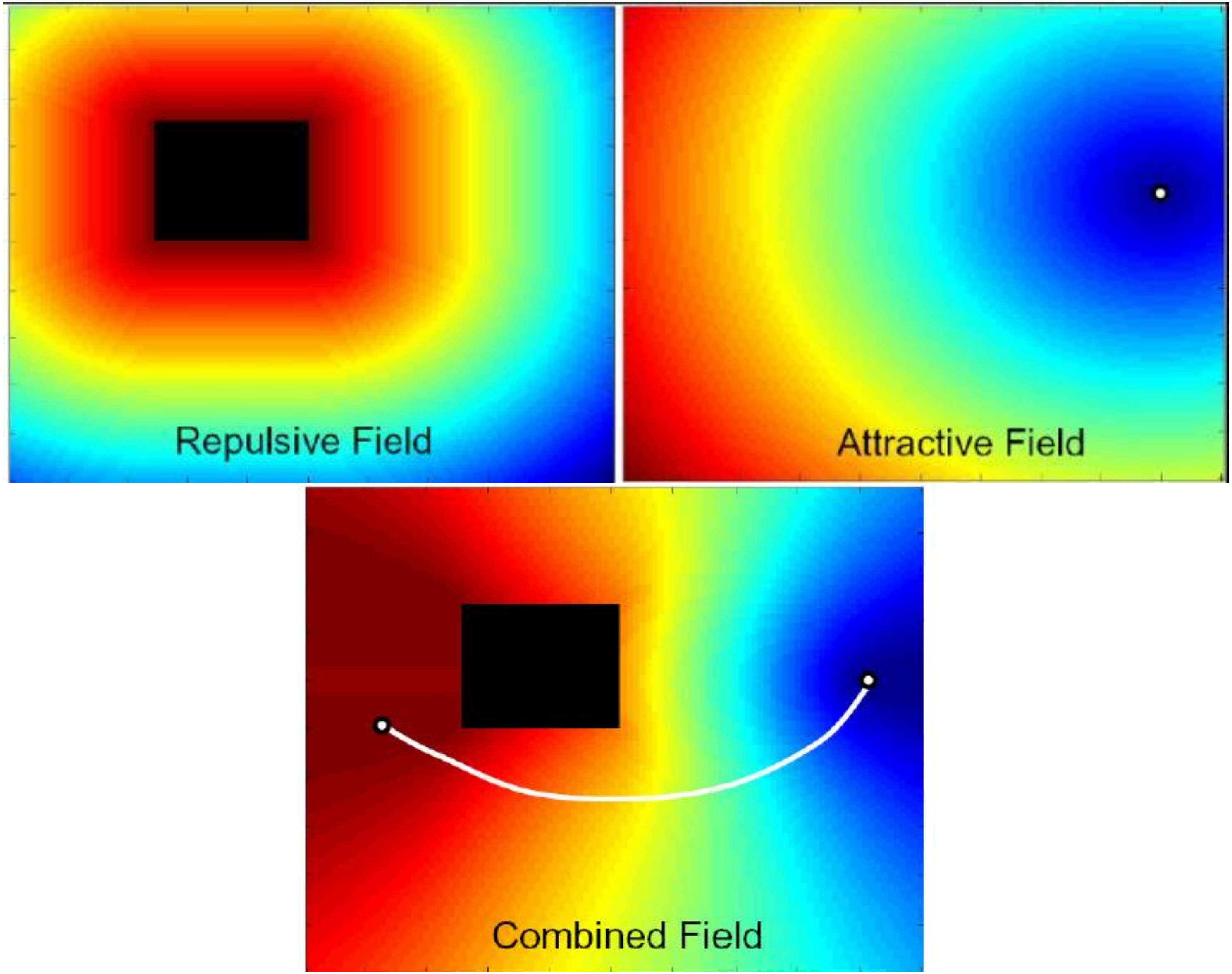
- $U_{goal}(q) = d^2(q, q_{goal})$
- $U_{obstacle}(q) = \frac{1}{d^2(q,obstacle)}$
- Potential: $U(q) = U_{goal}(q) + \lambda \sum U_{obstacle}(q)$
- Force field: $F(q) = -\nabla U(q)$
- Moving: take a small step in direction F

Relative strength of repulsive field

Negative gradient = "roll downhill"

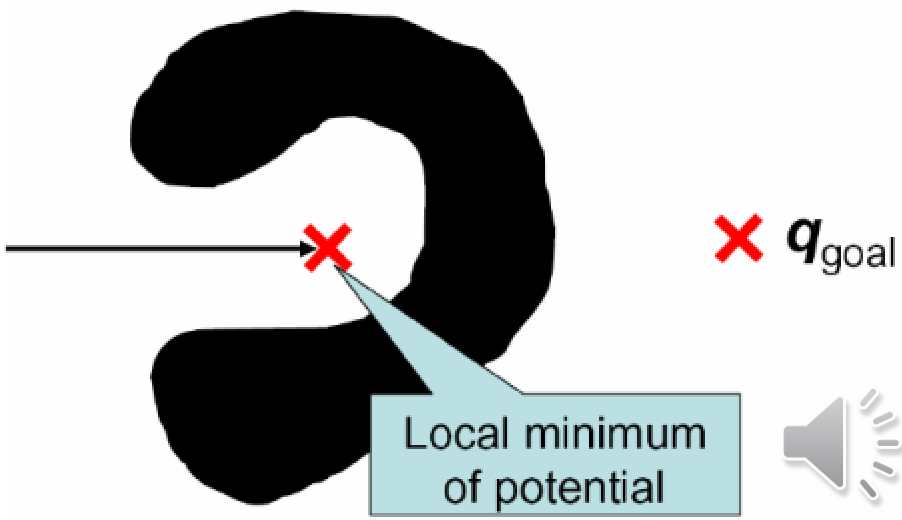


Potential Fields



Potential Fields

- Pro:
  - Spatial paths not pre-planned (generated in real time)
- Con:
  - Local minima
  - Fix with local suboptimal random walks
- Often used for local path planning



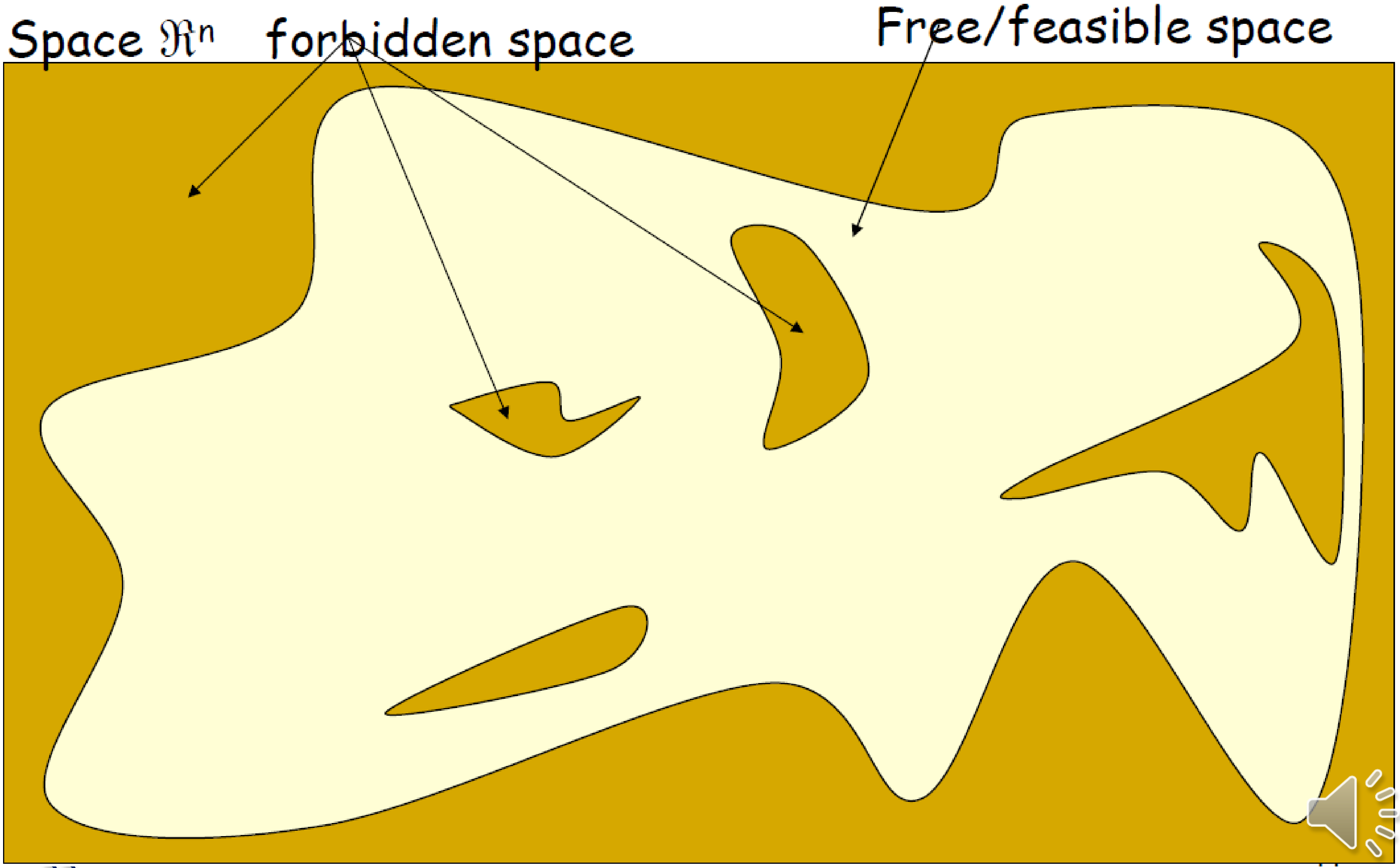
Sampling-based Approaches

- Completely describing and optimally exploring C-space is too hard in high dimensions
  - Instead, find a “good” sampling
- Typically, may need only a small number of samples to find a solution in a large space
- But, cannot detect that a path does not exist
- Need good sampling techniques
  - Uniform, near obstacles, in sparse areas

Probabilistic Roadmaps

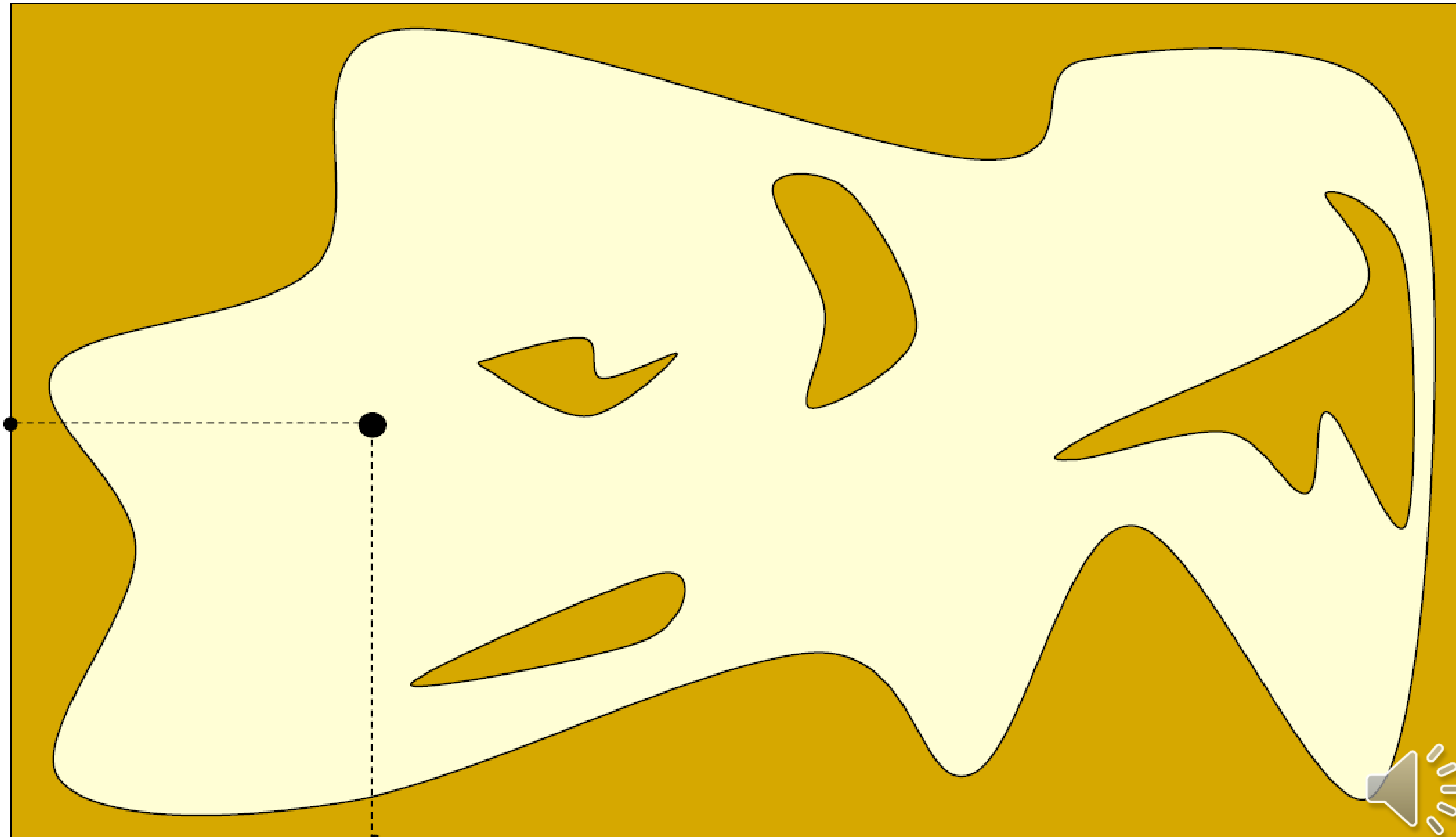
- Draw on previous ideas:
  - Identify special "landmark" points
  - Connect based on visibility
  - Plan paths through graph (e.g. A*)
- Build graph from random samples of free configurations
  - Probabilistic Roadmap (PRM)
- Learning phase (build map) vs query phase (plan)
- Multi-query planner

Probabilistic Roadmap



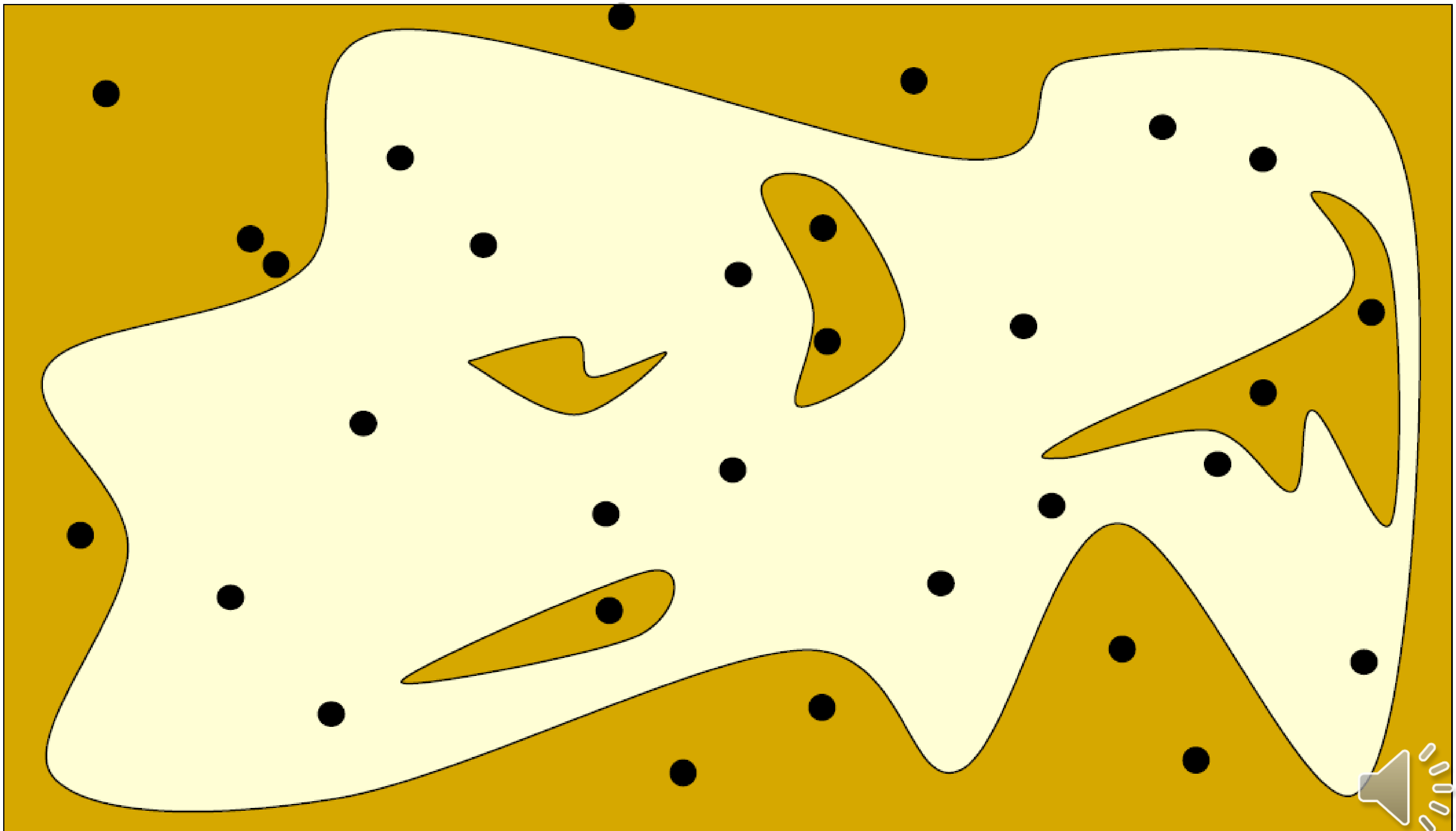
Probabilistic Roadmap

Configurations are sampled by picking coordinates at random



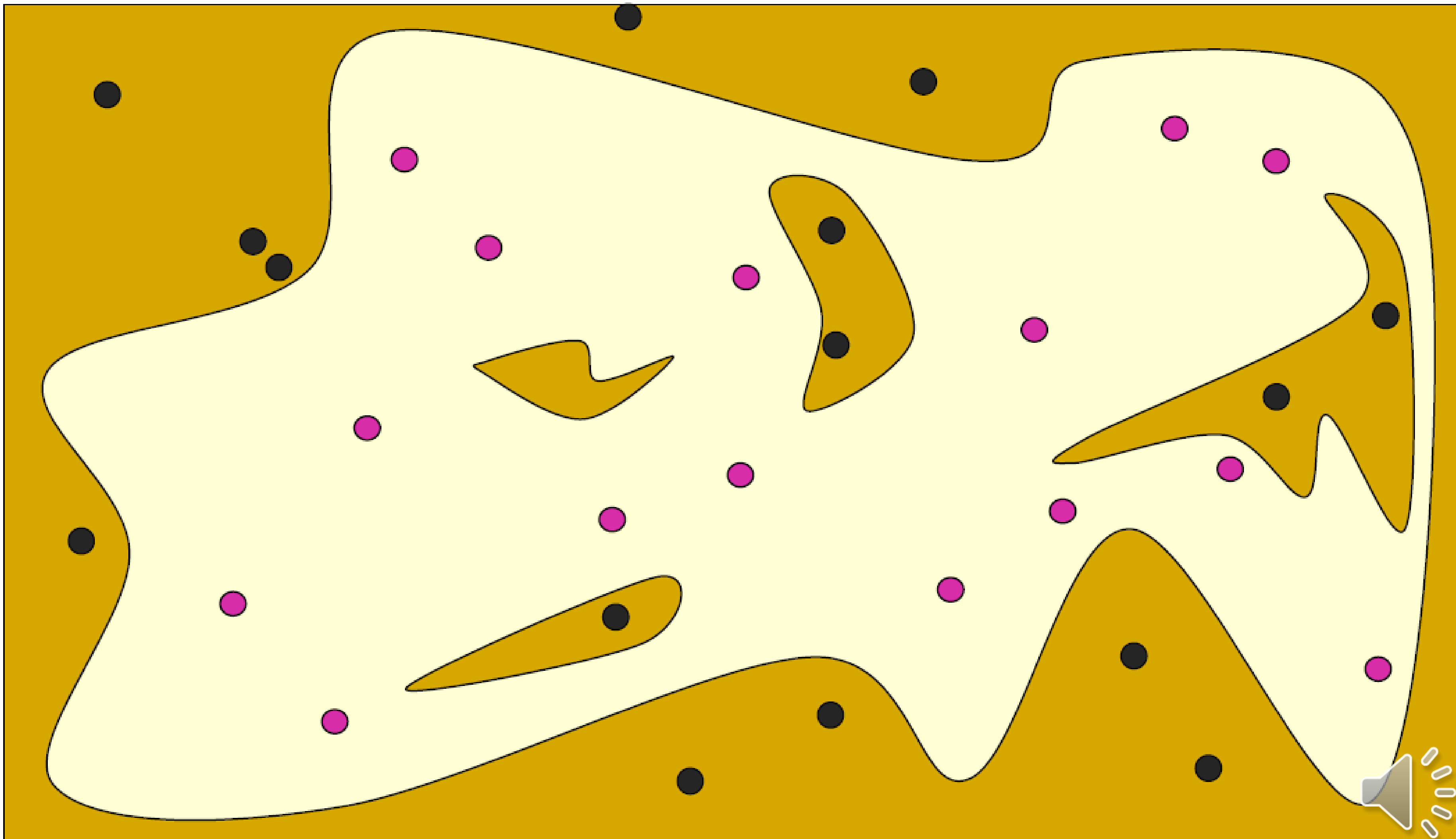
Probabilistic Roadmap

Configurations are sampled by picking coordinates at random



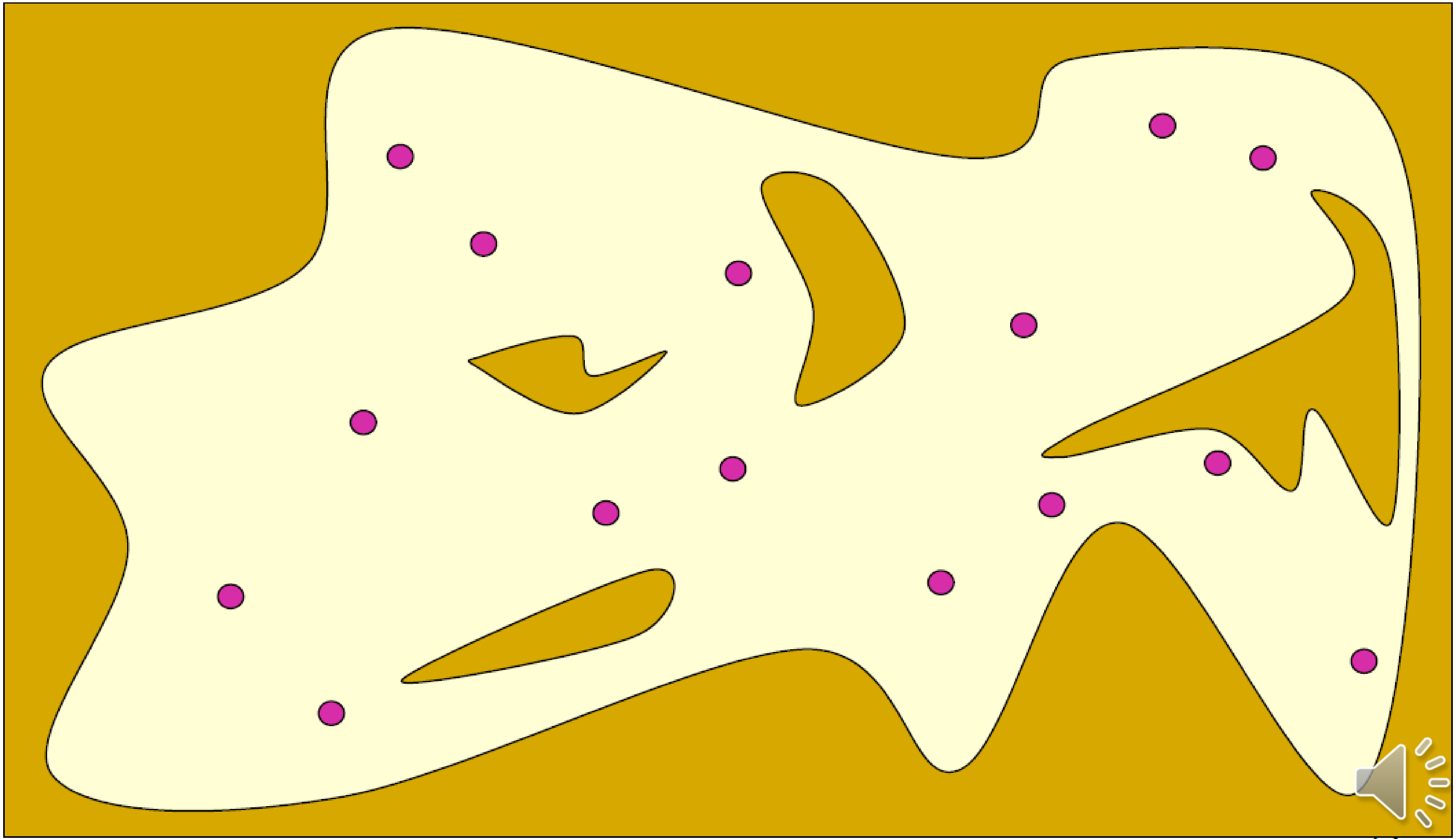
Probabilistic Roadmap

Sampled configurations are tested for collision



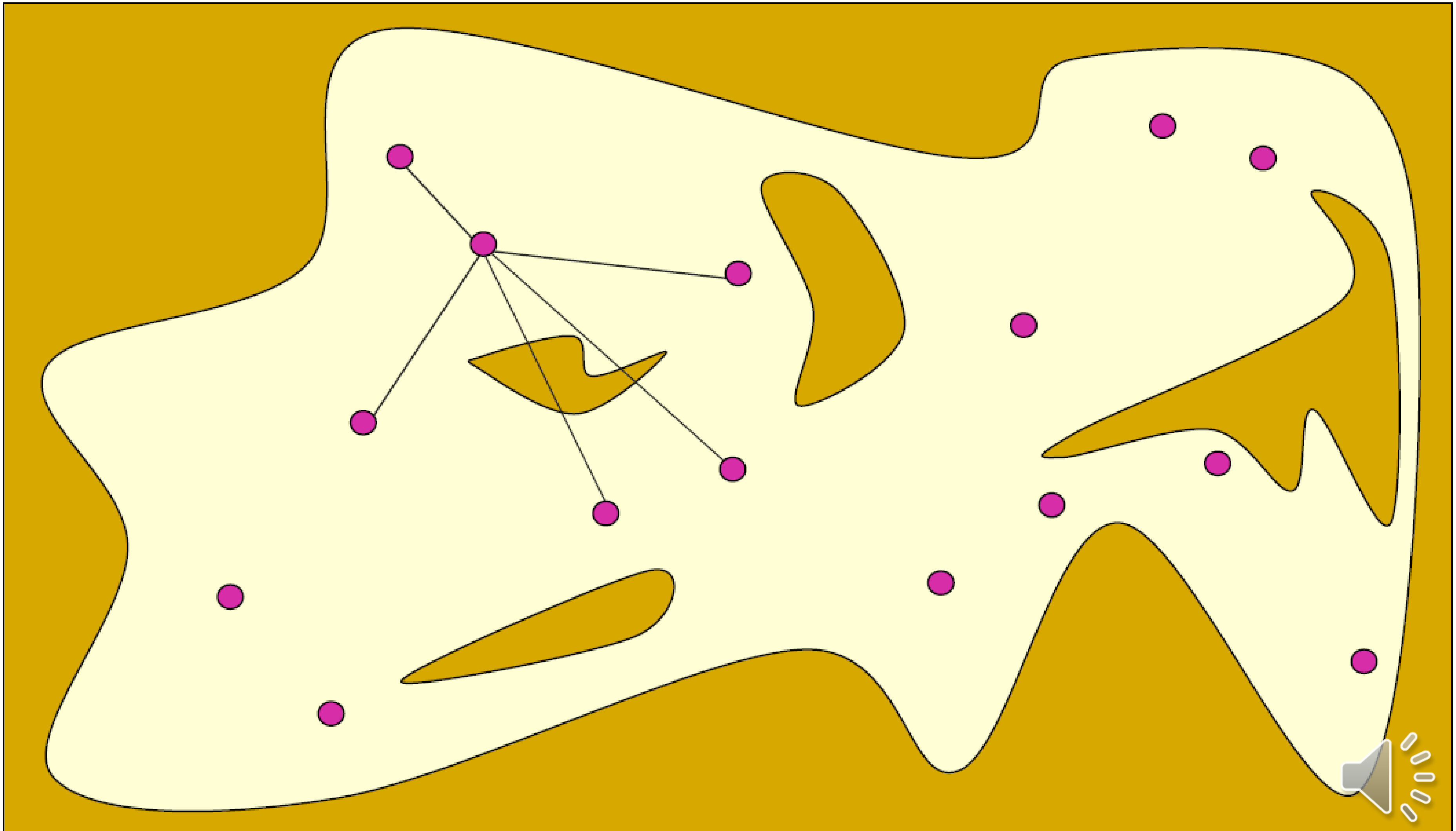
Probabilistic Roadmap

The collision-free configurations are retained as milestones



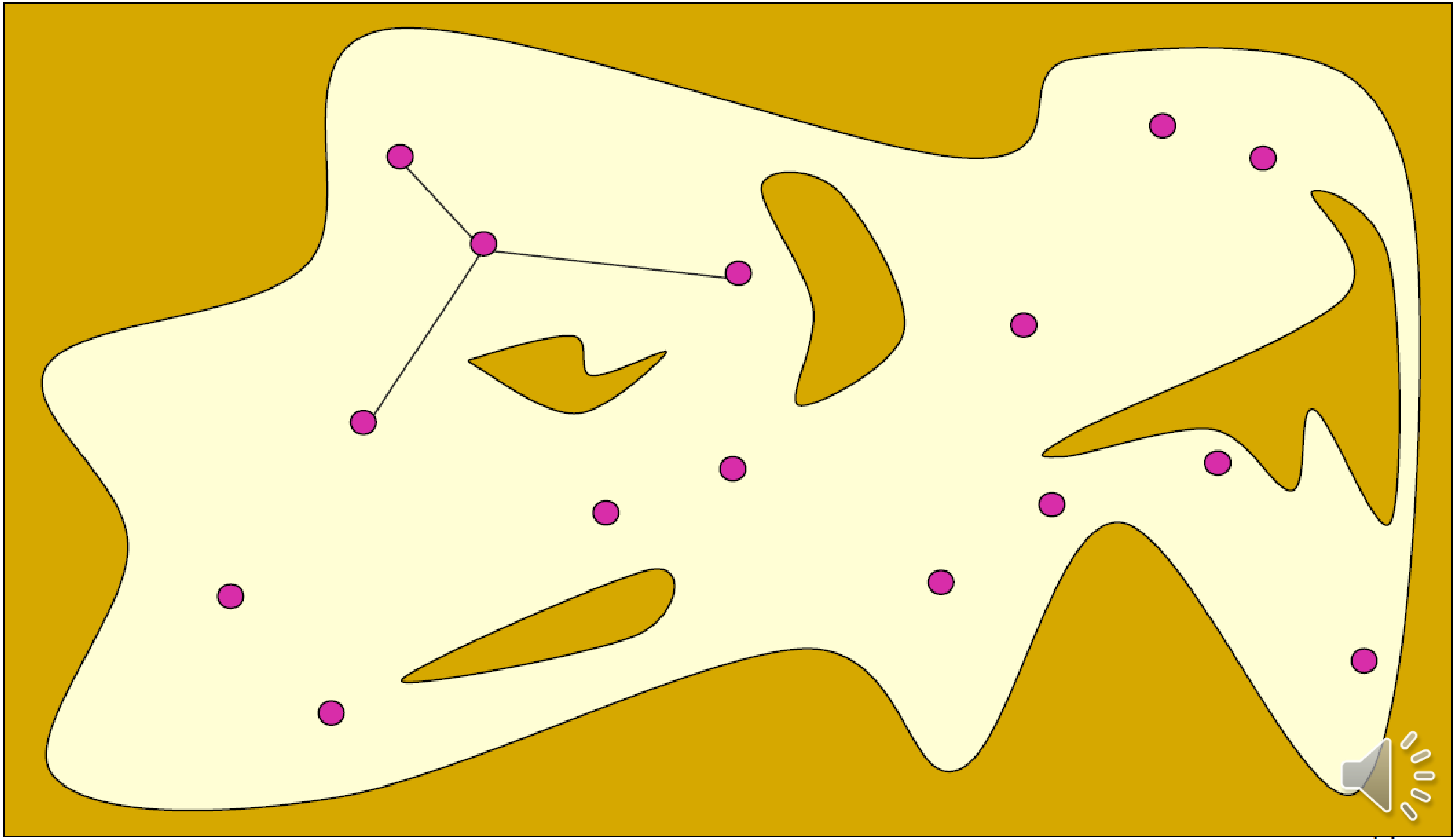
Probabilistic Roadmap

Each milestone is linked by straight paths to its nearest neighbors



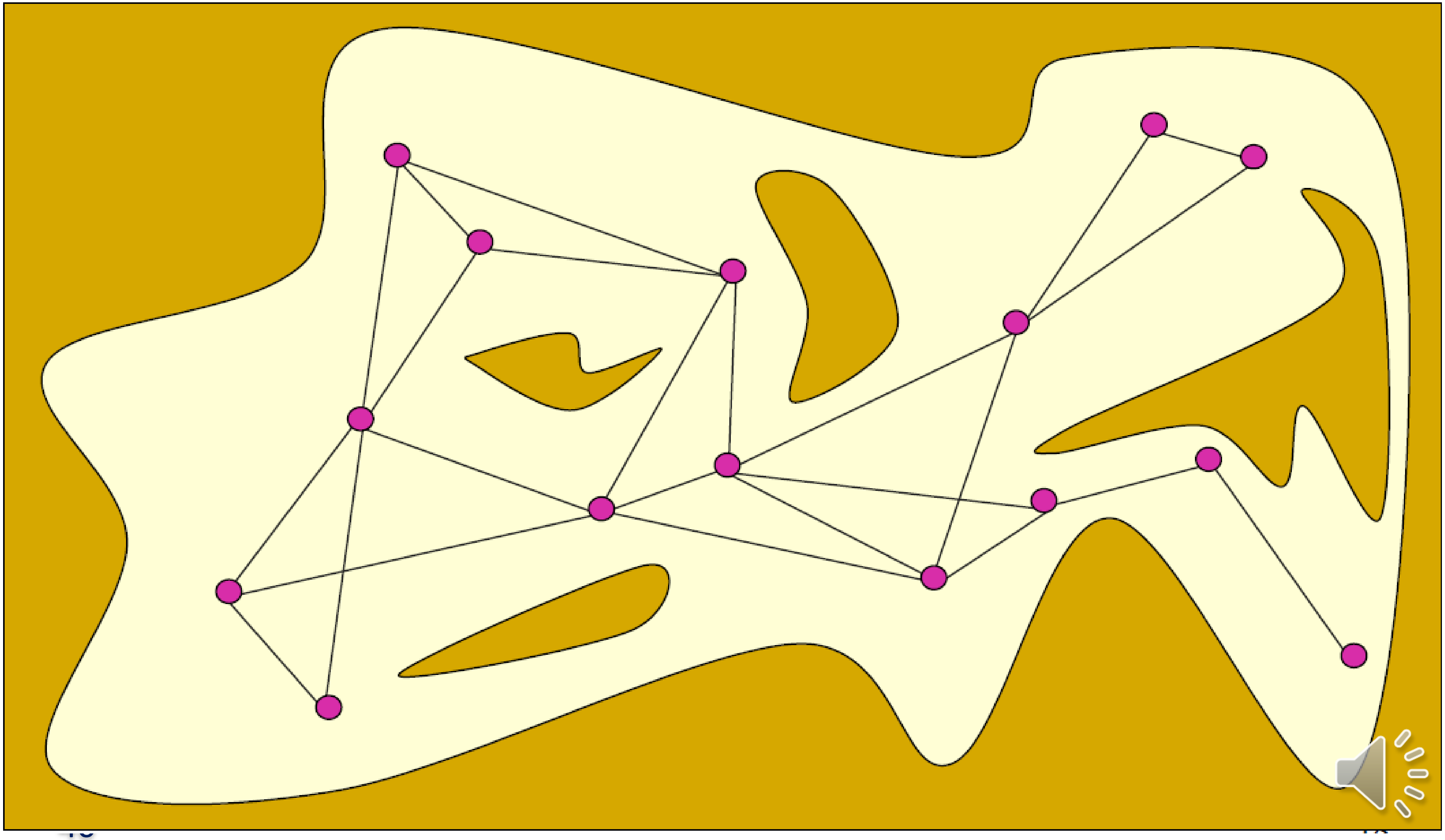
Probabilistic Roadmap

Each milestone is linked by straight paths to its nearest neighbors



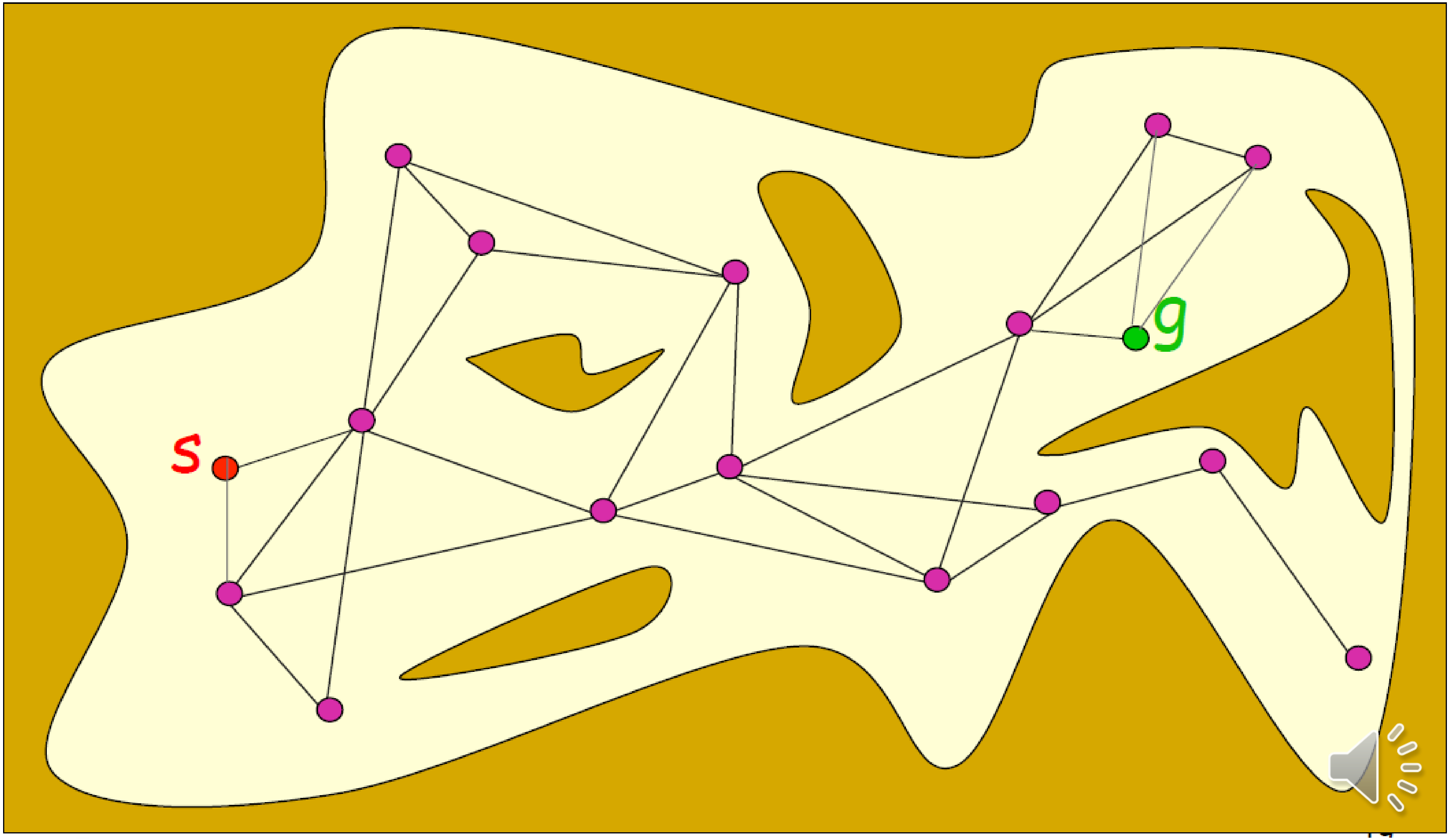
Probabilistic Roadmap

The collision-free links are retained as local paths to form the PRM



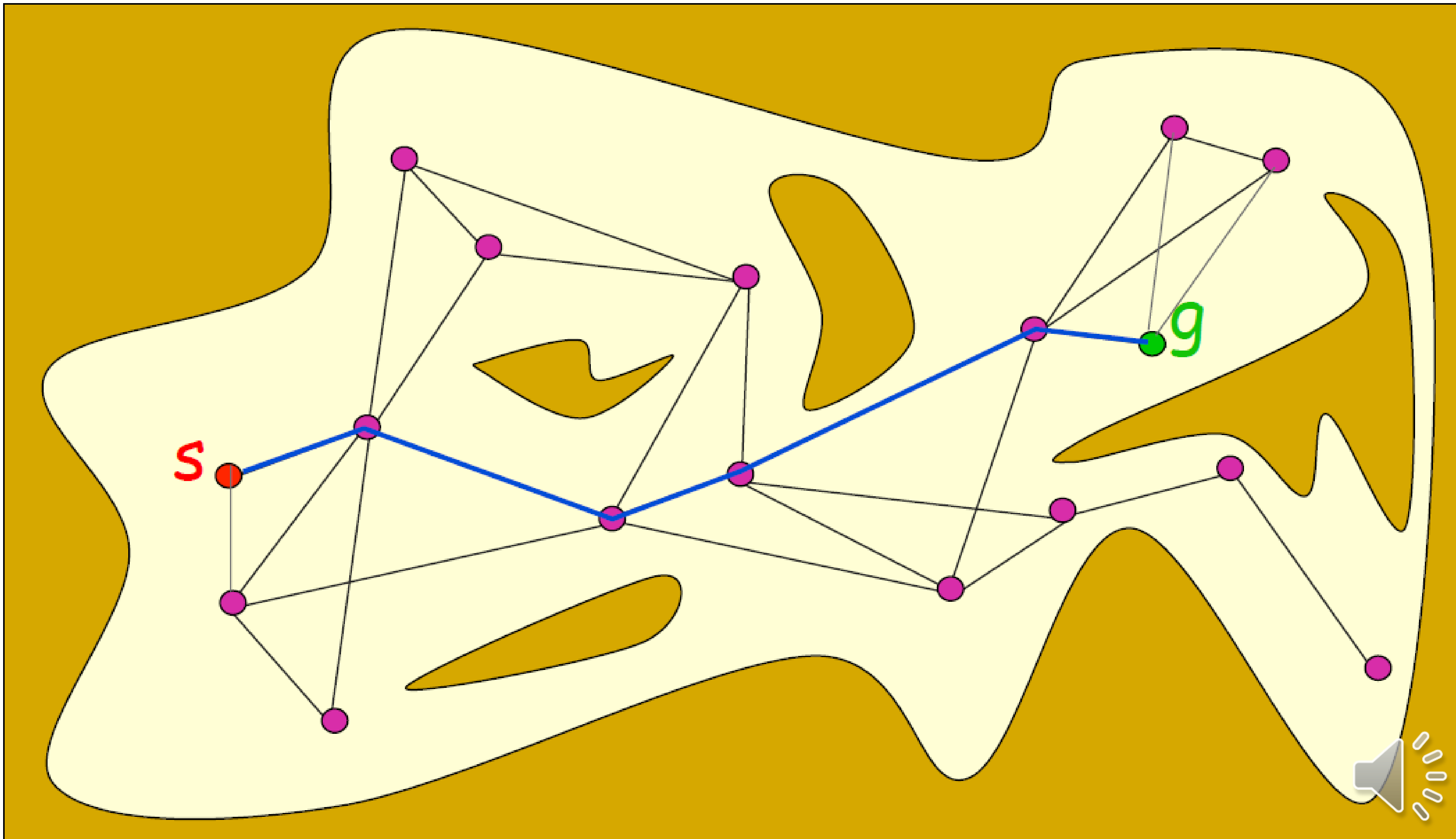
Probabilistic Roadmap

The start and goal configurations are included as milestones



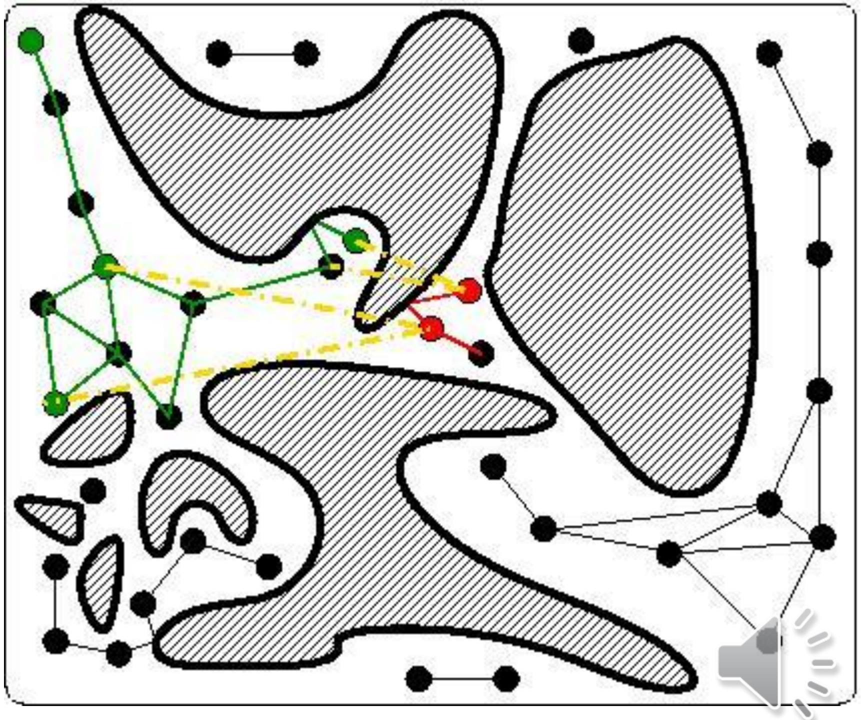
Probabilistic Roadmap

The PRM is searched for a path from s to g



PRM

- Pros:
  - Probabilistically complete: will find a solution if run for long enough
  - Multiple queries
- Cons:
  - Checking if nodes are connected/visible is expensive
  - What if non-holonomic – can't move directly between two points?
  - Expensive for single queries
  - Complete sampling?

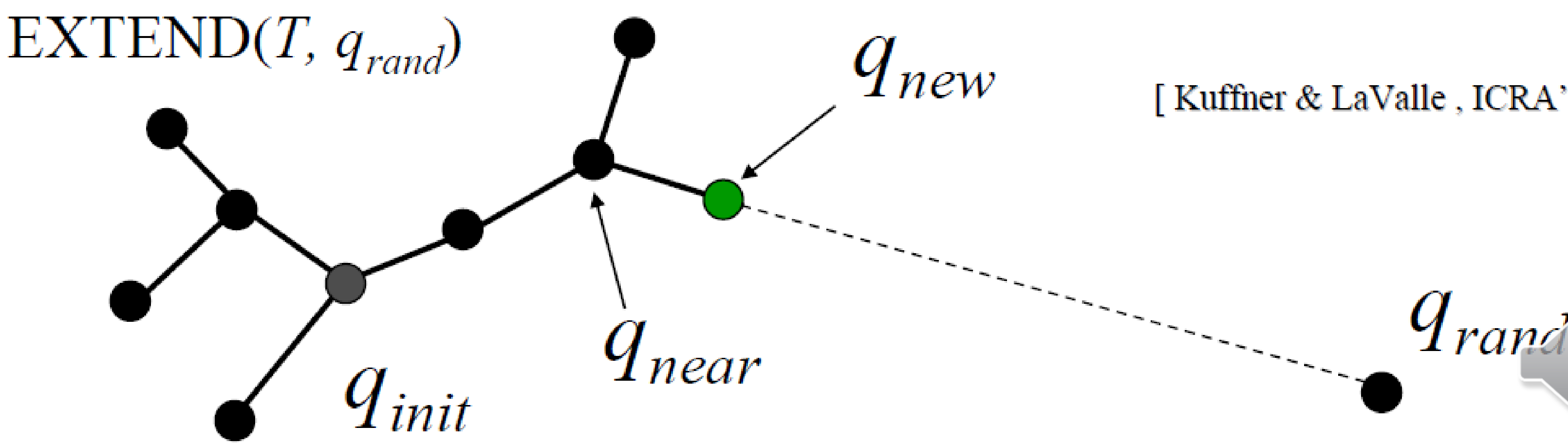


Rapidly-exploring Random Trees

- Incrementally construct a search tree that gradually improves resolution
- Dense covering of space
- Random samples
- Single query planner
  - But multi-query versions

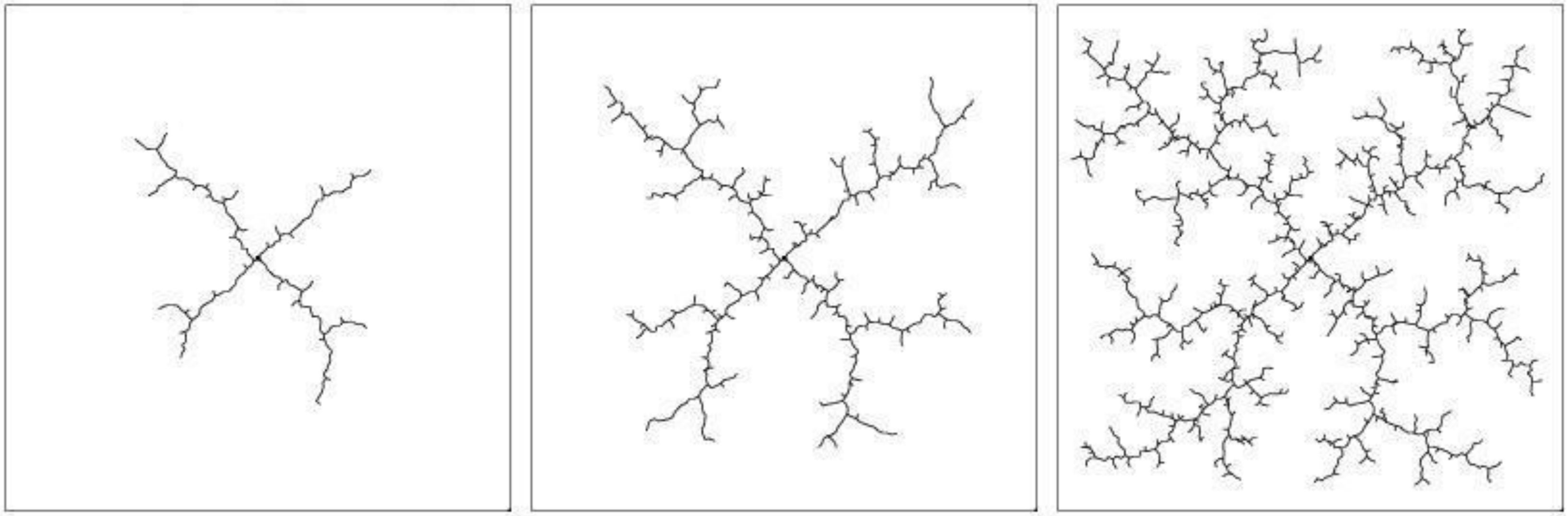
Rapidly-exploring Random Trees

- Start with initial configuration as root of tree
- Pick random state in C-space
- Find closest node in tree (approximate)
- Extend that node toward the state if possible
  - Collisions
  - Legal controls
- Repeat



[Kuffner & LaValle , ICRA'00]

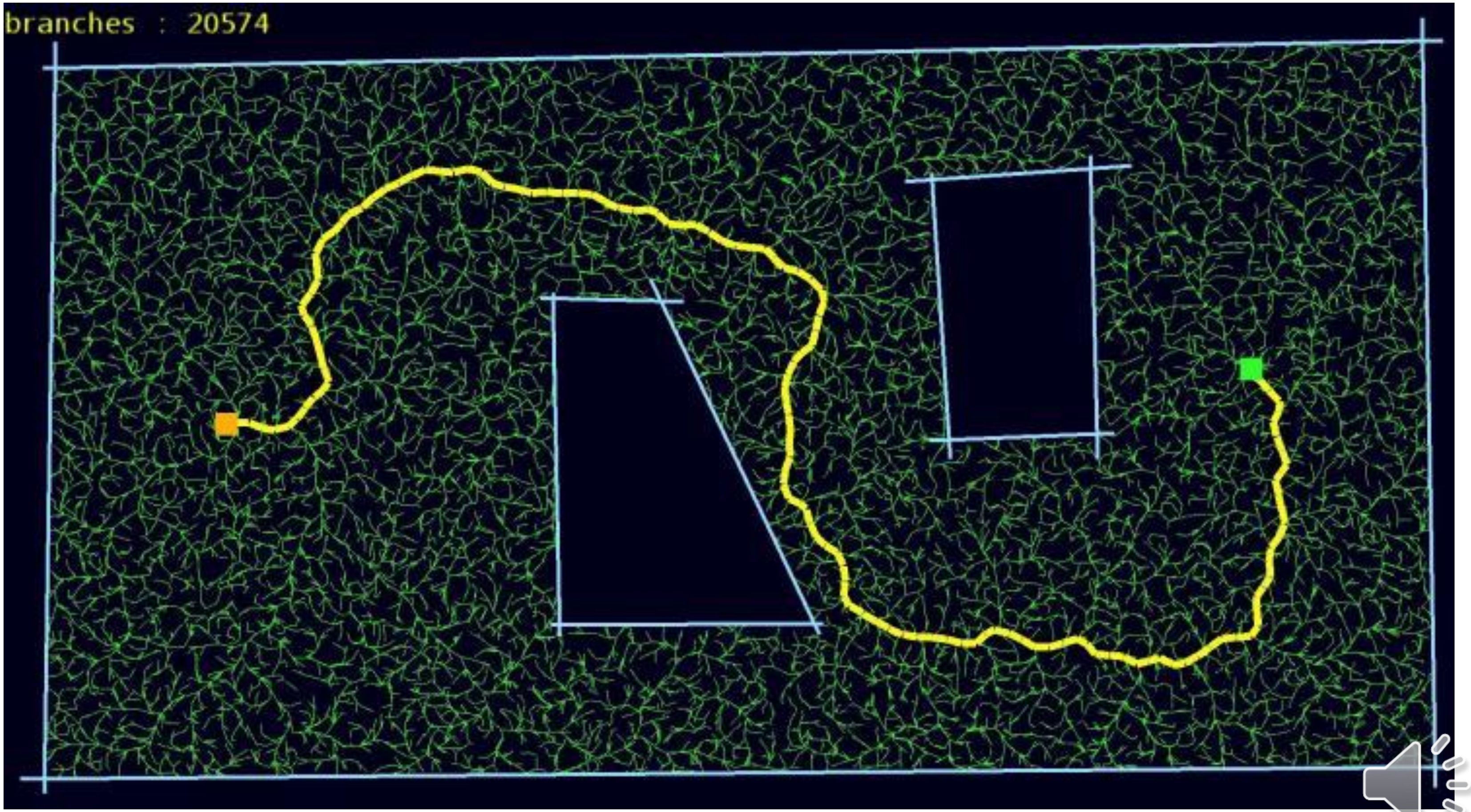
RRT



https://en.wikipedia.org/wiki/File:Rapidly-exploring_Random_Tree_(RRT)_500x373.gif

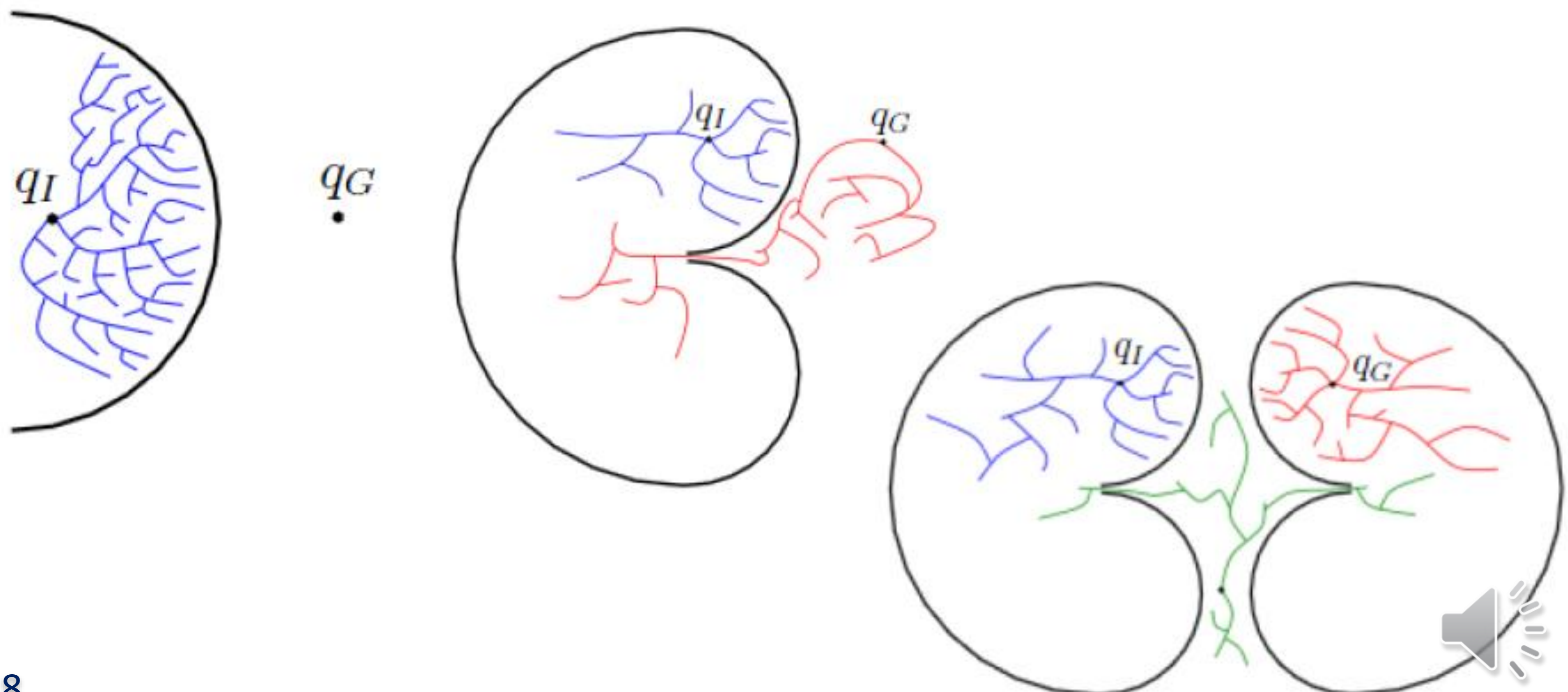
Rapidly-exploring Random Trees

- Often used with some form of smoothing



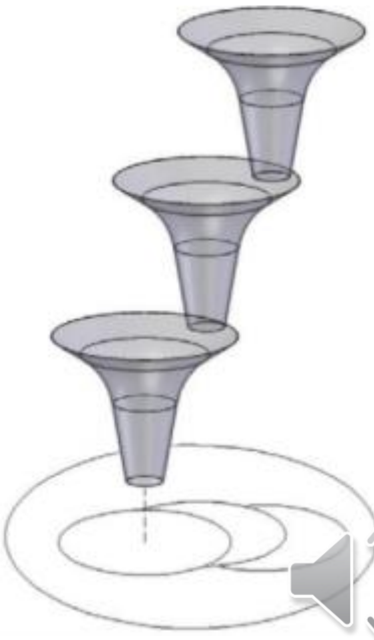
Rapidly-exploring Random Trees

- Planning around some obstacles can be easier from one direction than another (e.g. bug traps)
- Unidirectional, bidirectional, multi-directional

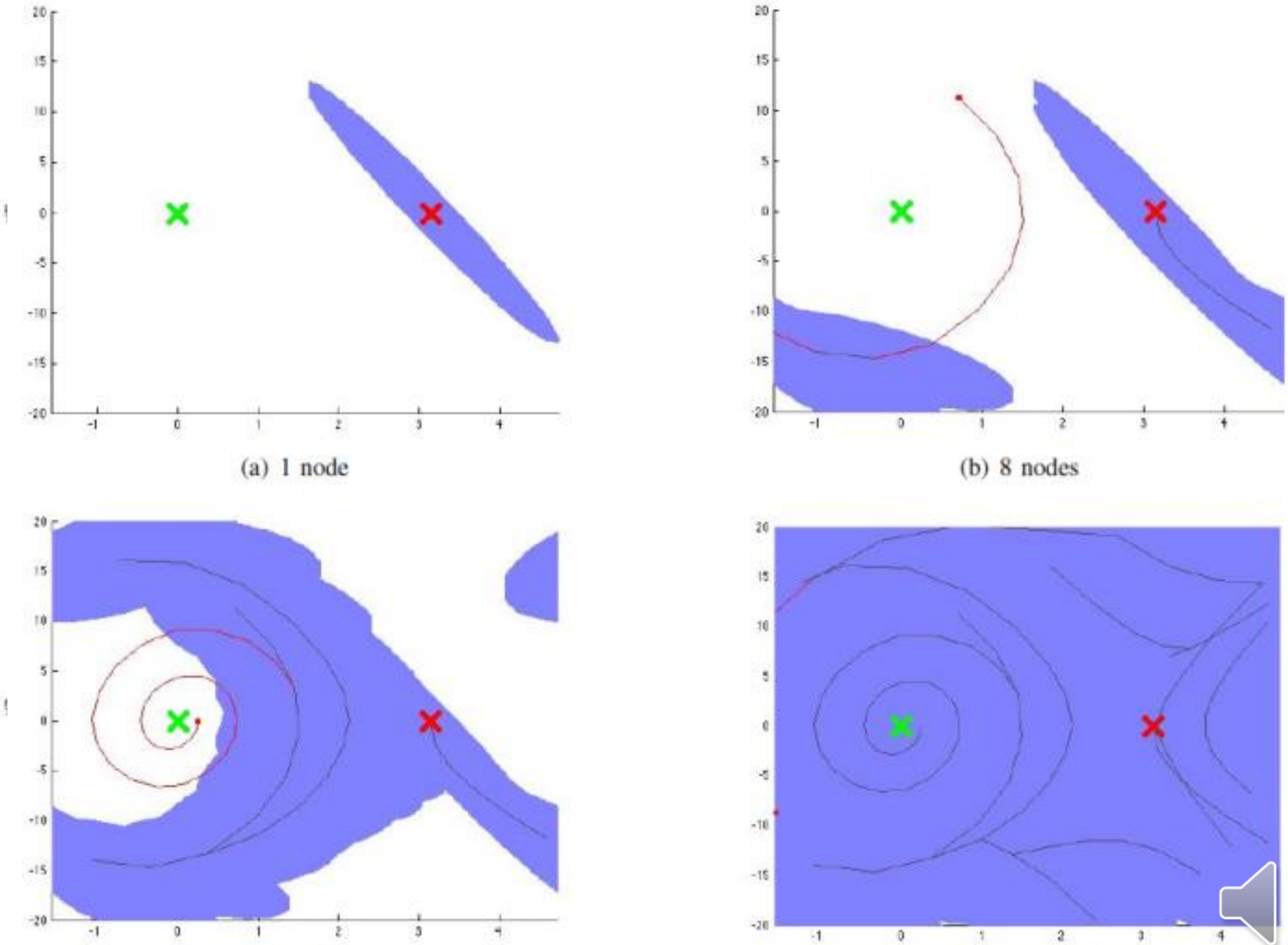


LQR-trees

- Add some notion of control and stability…
- Grow a randomised tree of stabilising LQR controllers
- Discard sampled points already in stabilised region



LQR-trees



(a) 1 node

(b) 8 nodes

(c) 24 nodes

(d) 104 nodes