16-350
Planning Techniques for Robotics

Planning Representations: Skeletonization- and Grid-based Graphs Explicit vs. Implicit Graphs

Maxim Likhachev

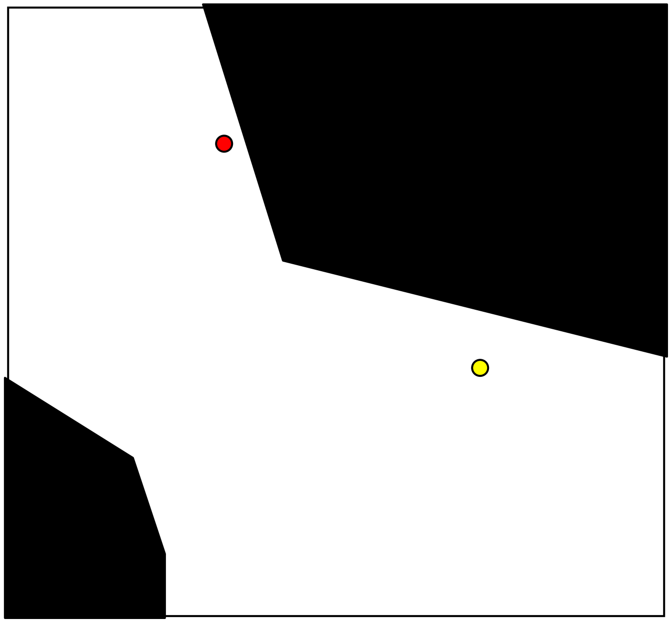
Robotics Institute

Carnegie Mellon University

2D Planning for Omnidirectional Point Robot

Planning for omnidirectional point robot:

What is $M^R = \langle x,y \rangle$
What is $M^W = \langle$*obstacle/free space*$\rangle$
What is $s^R_{current} = \langle x_{current}, y_{current} \rangle$
What is $s^W_{current}$ = *constant*
What is C = *Euclidean Distance*
What is $G = \langle x_{goal}, y_{goal} \rangle$



Planning as Graph Search Problem

1. Construct a graph representing the planning problem
2. Search the graph for a (hopefully, close-to-optimal) path

The two steps above are often interleaved

Planning as Graph Search Problem

1. Construct a graph representing the planning problem

This class

2. Search the graph for a (hopefully, close-to-optimal) path

Next class

The two steps above are often interleaved

More on this in this & later classes

2D Planning for Omnidirectional Point Robot

Planning for omnidirectional point robot:

What is M^R = <x,y>

What is M^W = <*obstacle/free space*>

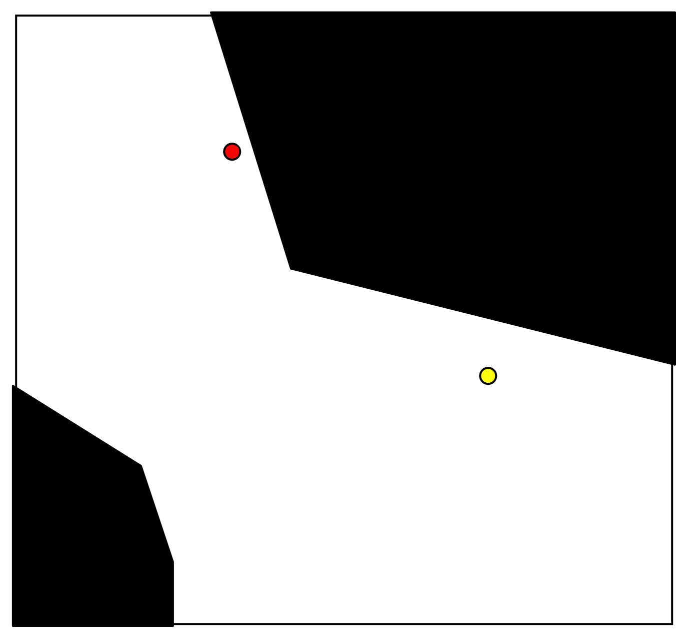
What is $s^R_{current}$ = <$x_{current}, y_{current}$>

What is $s^W_{current}$ = *constant*

What is C = *Euclidean Distance*

What is G = <x_{goal}, y_{goal}>

Any ideas on how to construct a graph for planning?



Two Classes of Graph Construction Methods

- Skeletonization

  -Visibility graphs

  -Voronoi diagrams

  - Probabilistic roadmaps

- Cell decomposition

  - X-connected grids

  - lattice-based graphs

Two Classes of Graph Construction Methods

- Skeletonization

  -Visibility graphs

  -Voronoi diagrams

  - Probabilistic roadmaps

Will be covered in later classes

- Cell decomposition

  - X-connected grids

  - lattice-based graphs

Two Classes of Graph Construction Methods

- Skeletonization

  -Visibility graphs

  -Voronoi diagrams

  - Probabilistic roadmaps

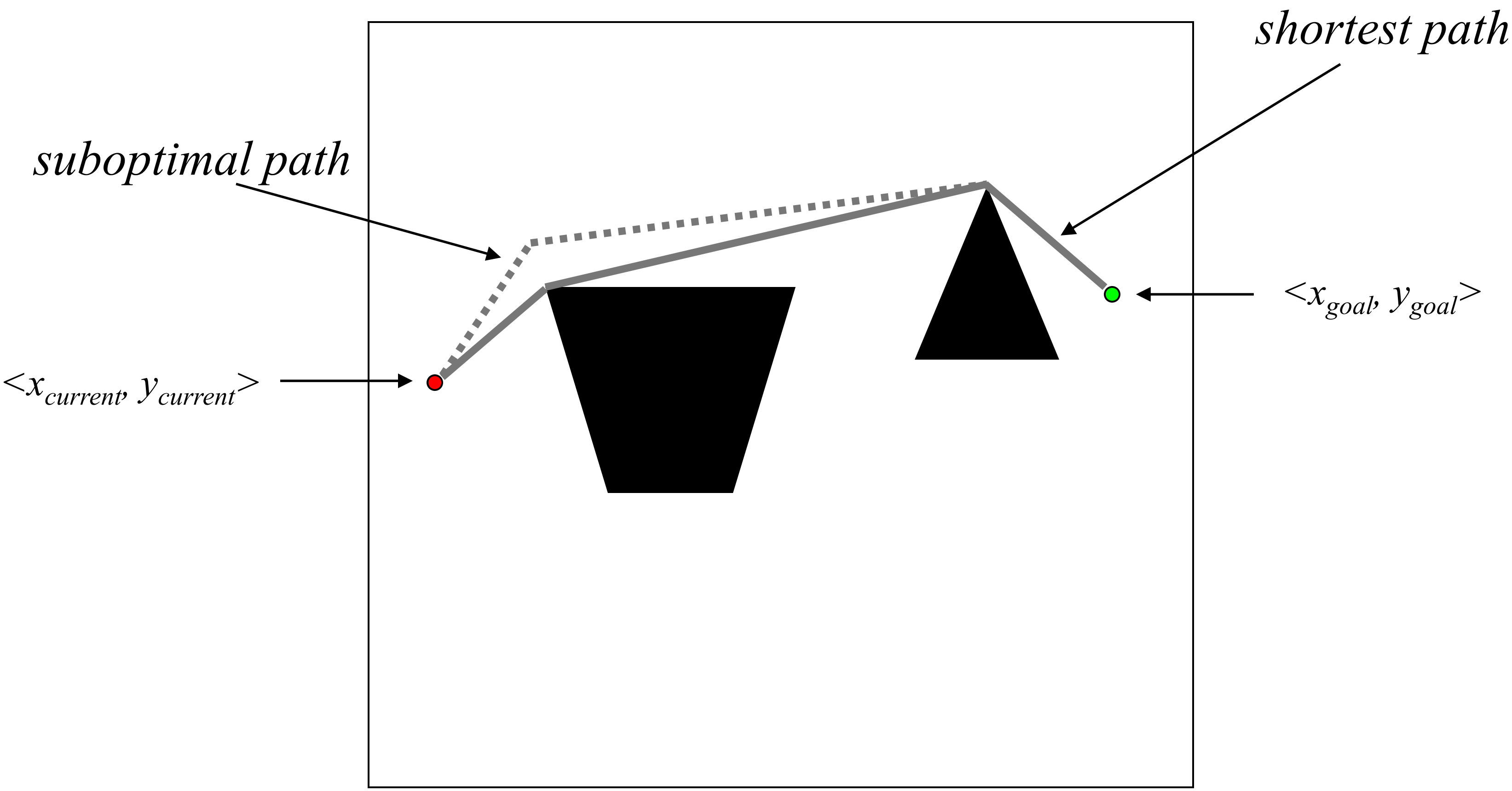
- Cell decomposition

  - X-connected grids

  - lattice-based graphs

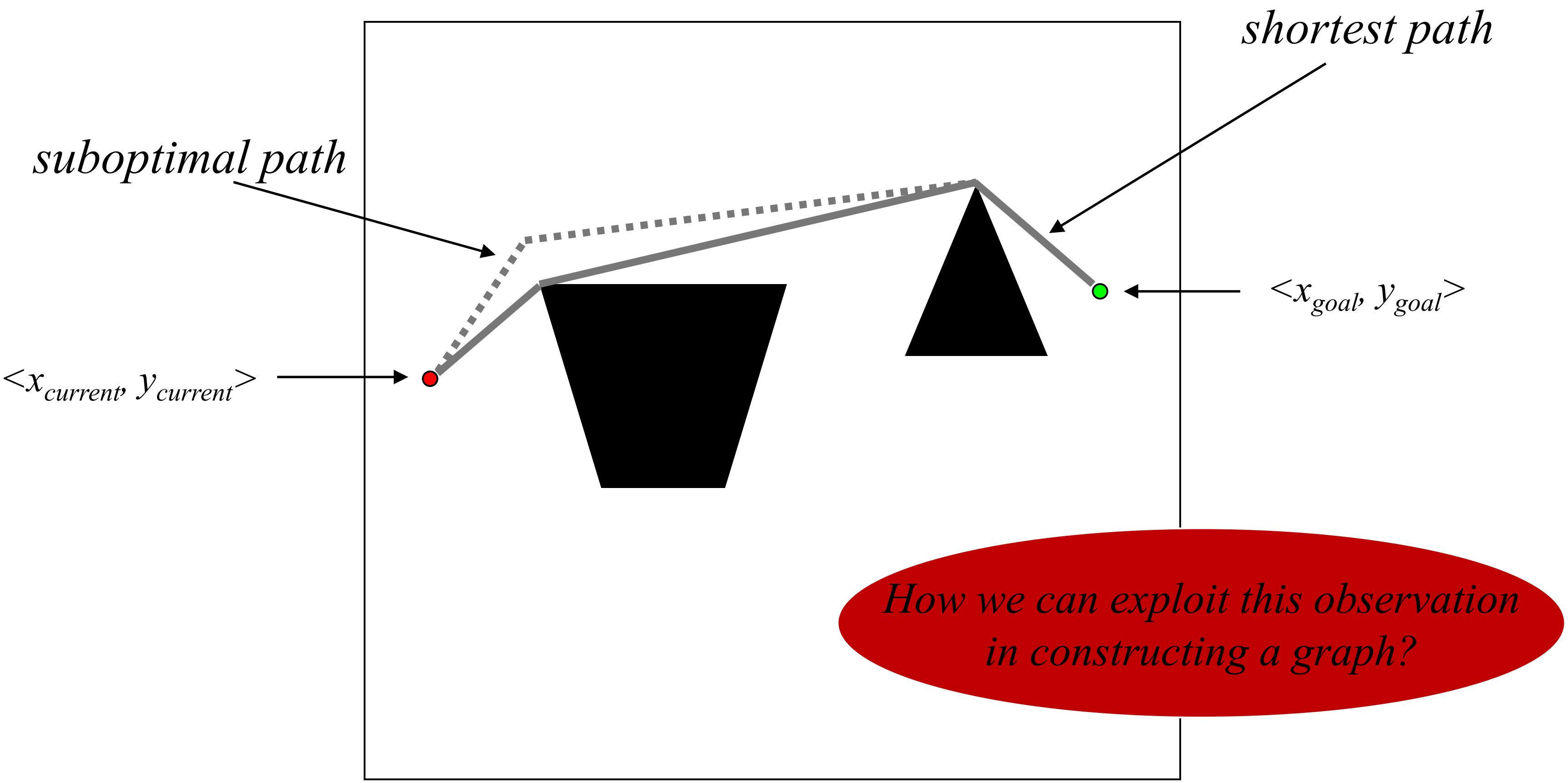
Skeletonization-based Graphs

- Visibility Graphs [Wesley & Lozano-Perez '79]
  - based on idea that *the shortest path consists of obstacle-free straight line segments connecting all obstacle vertices and start and goal*



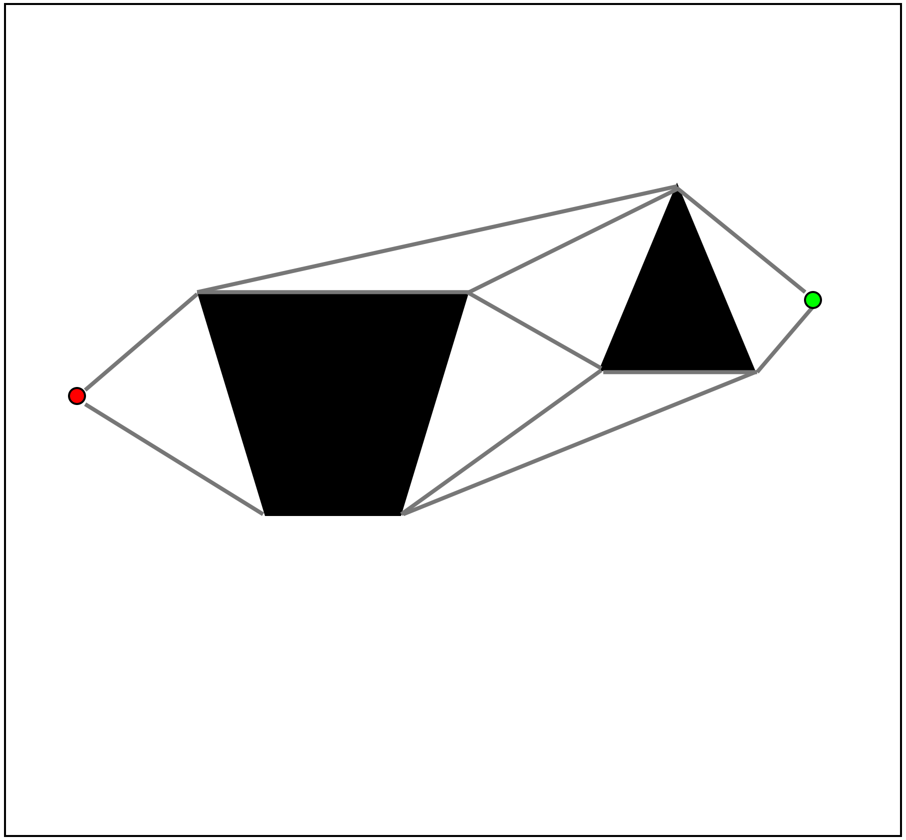
Skeletonization-based Graphs

- Visibility Graphs [Wesley & Lozano-Perez ’79]
  - based on idea that *the shortest path consists of obstacle-free straight line segments connecting all obstacle vertices and start and goal*



Skeletonization-based Graphs

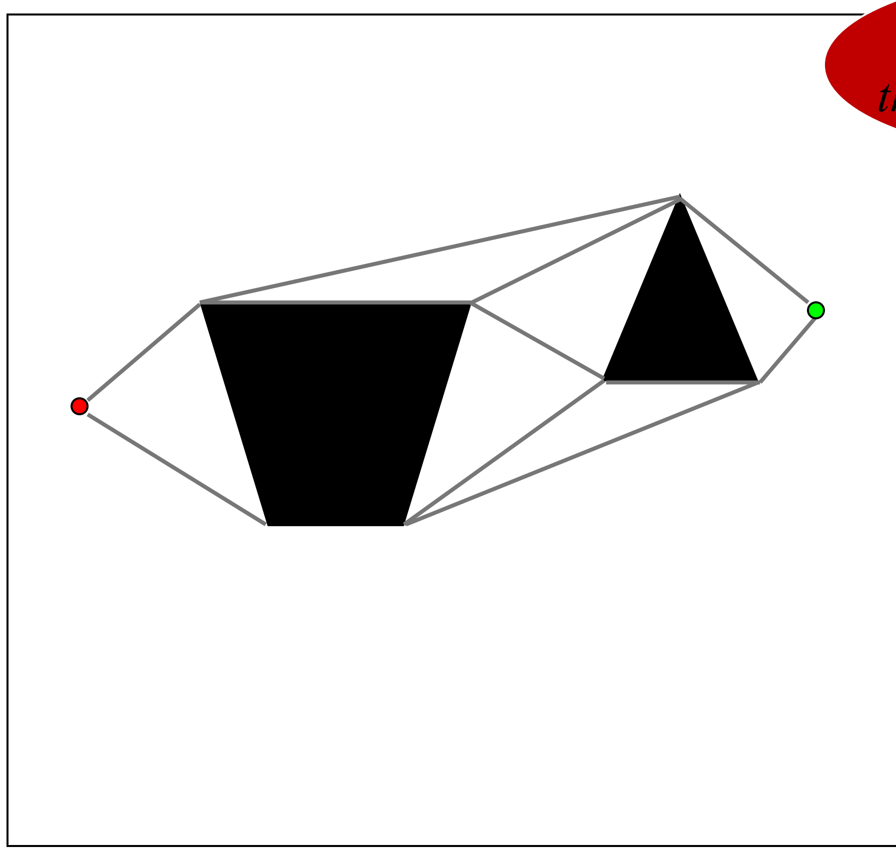
- Visibility Graphs [Wesley & Lozano-Perez '79]
  - construct a graph by connecting all vertices, start and goal by obstacle-free straight line segments (graph is $O(n^2)$, where n - # of vert.)



Skeletonization-based Graphs

- Visibility Graphs [Wesley & Lozano-Perez '79]
  - construct a graph by connecting all vertices, start and goal by obstacle-free straight line segments (graph is $O(n^2)$, where n - # of vert.)

Disadvantages of the Visibility Graphs?



Skeletonization-based Graphs

- Visibility Graphs
  - advantages:
    - independent of the size of the environment
  - disadvantages:
    - path is too close to obstacles
    - hard to deal with the cost function that is not distance
    - hard to deal with non-polygonal obstacles
    - hard to maintain the polygonal representation of obstacles
    - can be expensive in spaces higher than 2D

Two Classes of Graph Construction Methods

- Skeletonization

  -Visibility graphs

  -Voronoi diagrams

  - Probabilistic roadmaps

- Cell decomposition

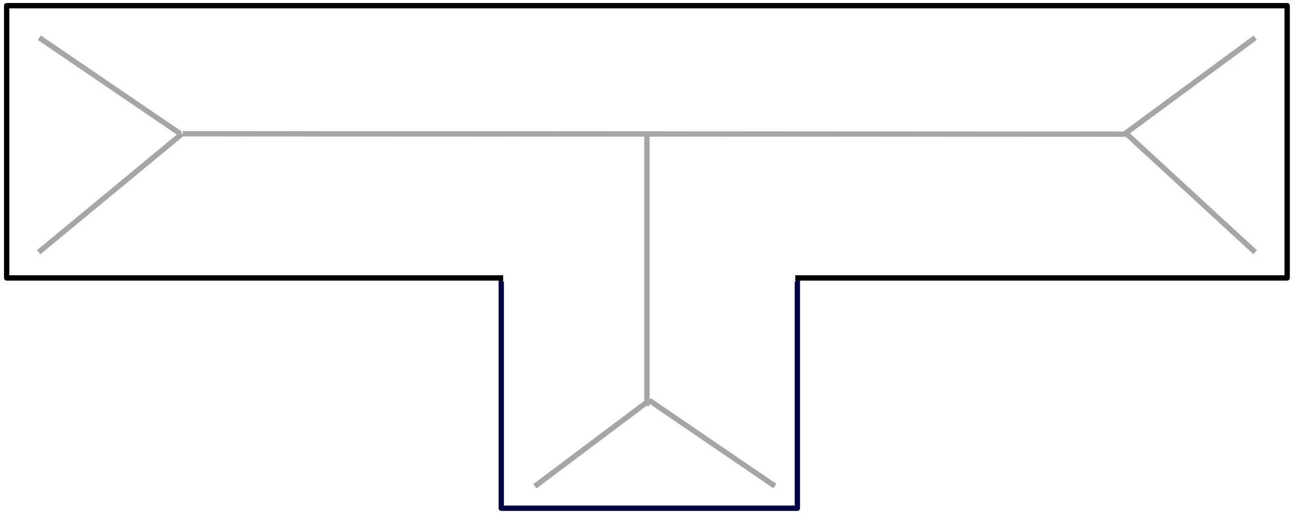
  - X-connected grids

  - lattice-based graphs

Skeletonization-based Graphs

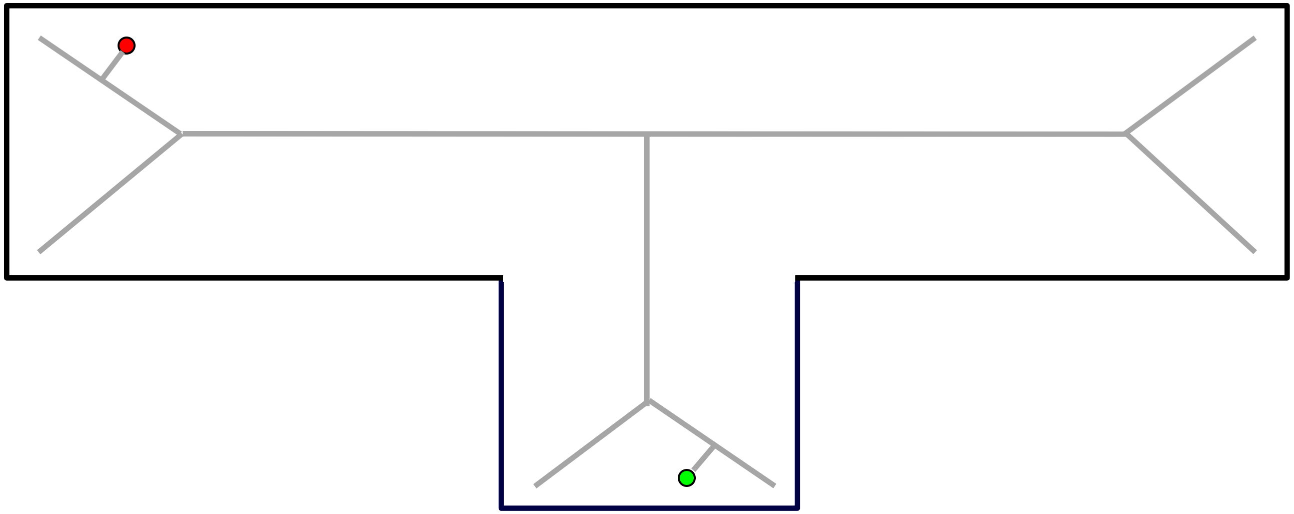
- Voronoi diagram [Rowat '79]
  - set of all points that are equidistant to two nearest obstacles

  (can be computed O (n log n), where n - # of points that represent obstacles)



Skeletonization-based Graphs

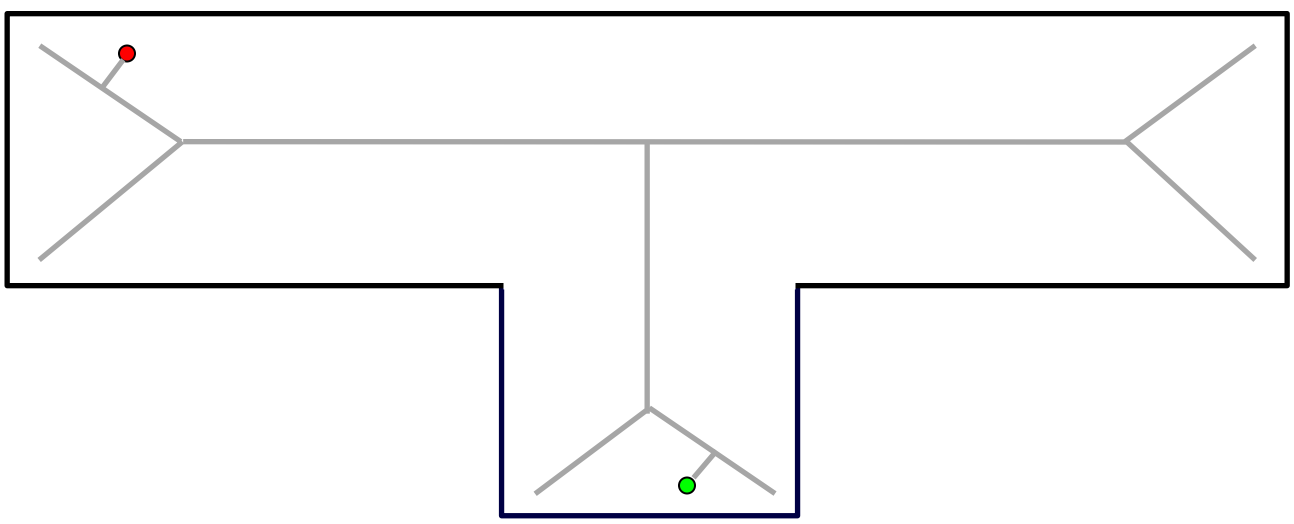
- Voronoi diagram-based graph
  - Edges: Boundaries in Voronoi diagram
  - Vertices: Intersection of boundaries
  - Add start and goal vertices
  - Add edges that correspond to:
    - shortest path segment from start to the nearest segment on the Voronoi diagram
    - shortest path segment from goal to the nearest segment on the Voronoi diagram



Skeletonization-based Graphs

- Voronoi diagram-based graph
  - Edges: Boundaries in Voronoi diagram
  - Vertices: Intersection of boundaries
  - Add start and goal vertices
  - Add edges that correspond to:
    - shortest path segment from start to the nearest segment on the Voronoi diagram
    - shortest path segment from goal to the nearest segment on the Voronoi diagram

Disadvantages of the Voronoi diagram-based Graphs?



Skeletonization-based Graphs

- Voronoi diagram-based graph
  - advantages:
    - tends to stay away from obstacles
    - independent of the size of the environment
    - can work with any obstacles represented as set of points
  - disadvantages:
    - can result in highly suboptimal paths
    - hard to deal with the cost function that is not distance
    - hard to use/maintain beyond 2D

Two Classes of Graph Construction Methods

- Skeletonization

  -Visibility graphs

  -Voronoi diagrams

  - Probabilistic roadmaps

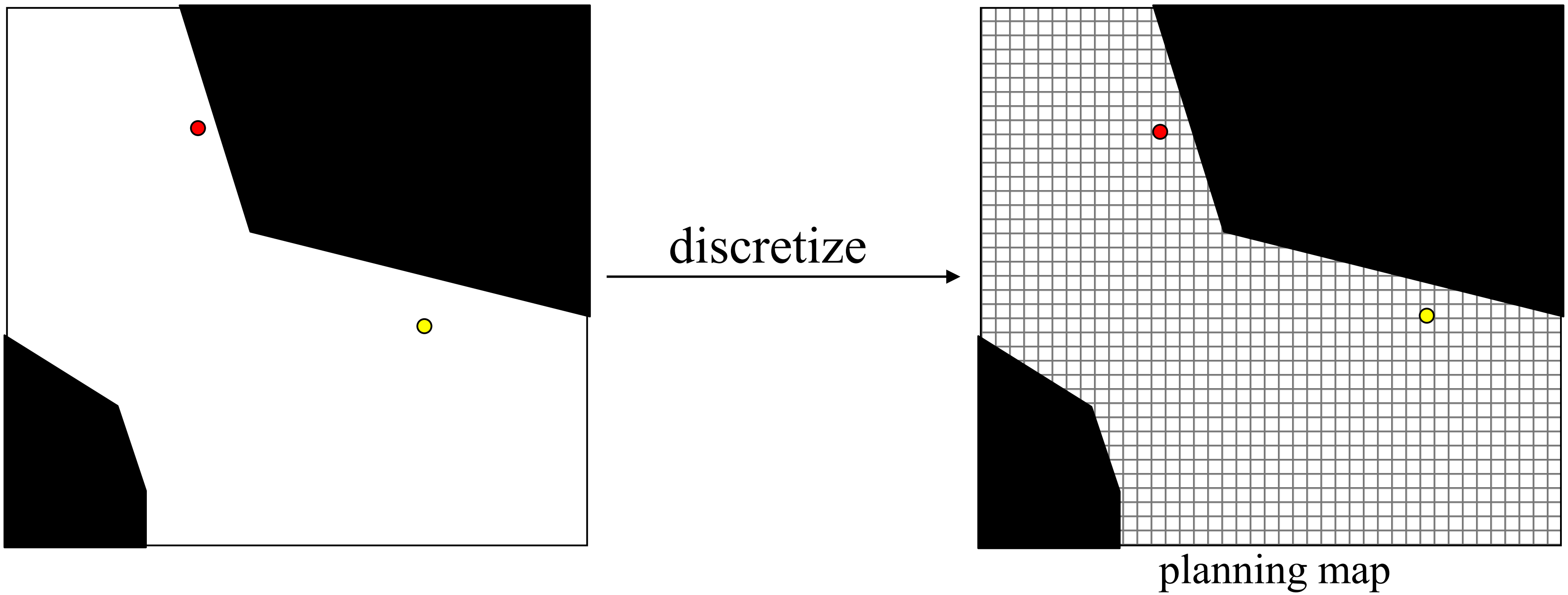
- Cell decomposition

  - X-connected grids

  - lattice-based graphs

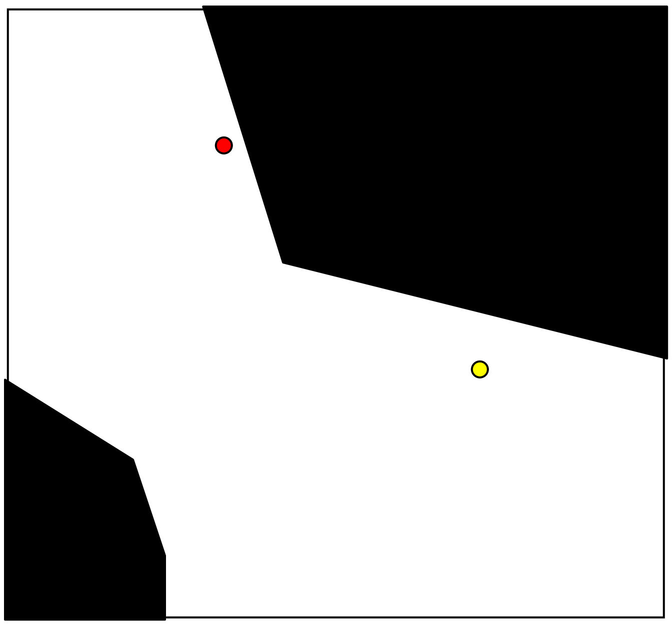
Grid-based Graphs

- Approximate Cell Decomposition:
    - overlay uniform grid (discretize)

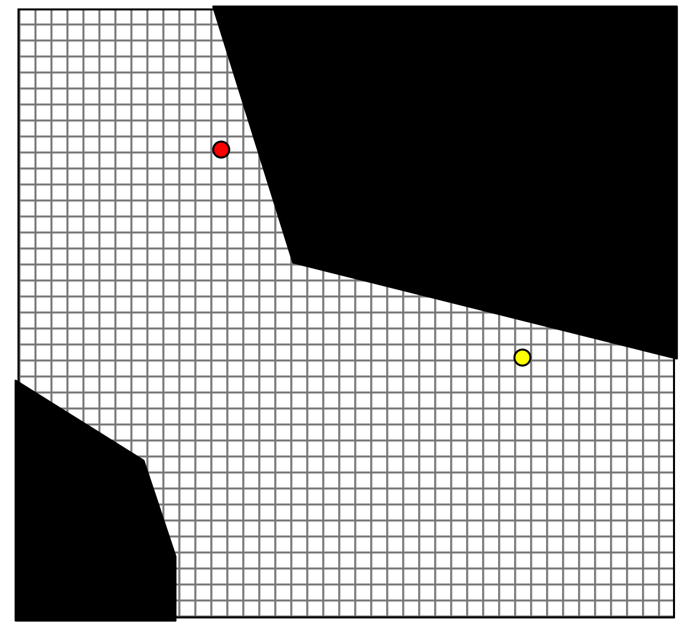


Grid-based Graphs

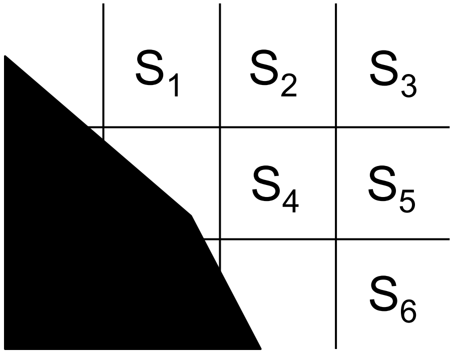
- Approximate Cell Decomposition:
  - construct a graph



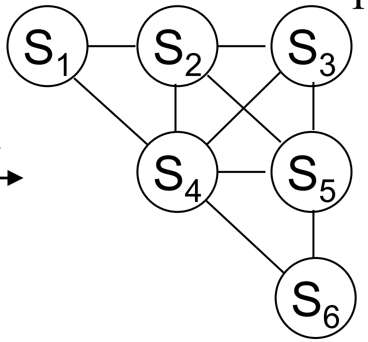
discretize



planning map



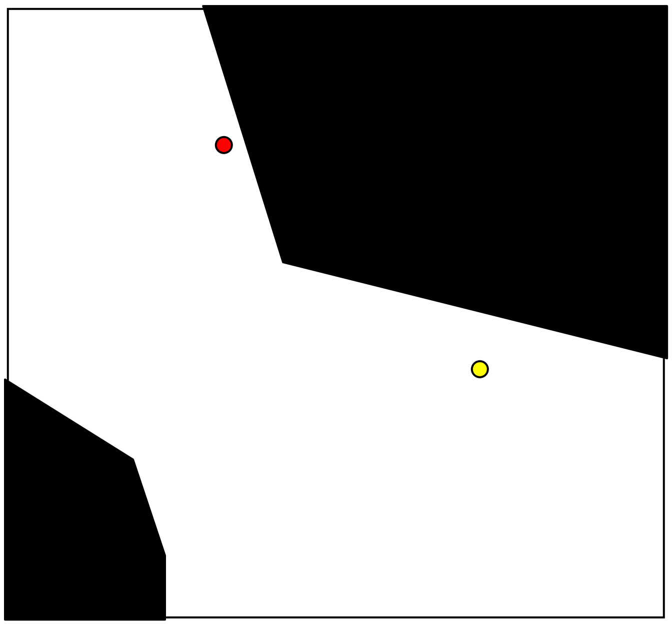
convert into a graph



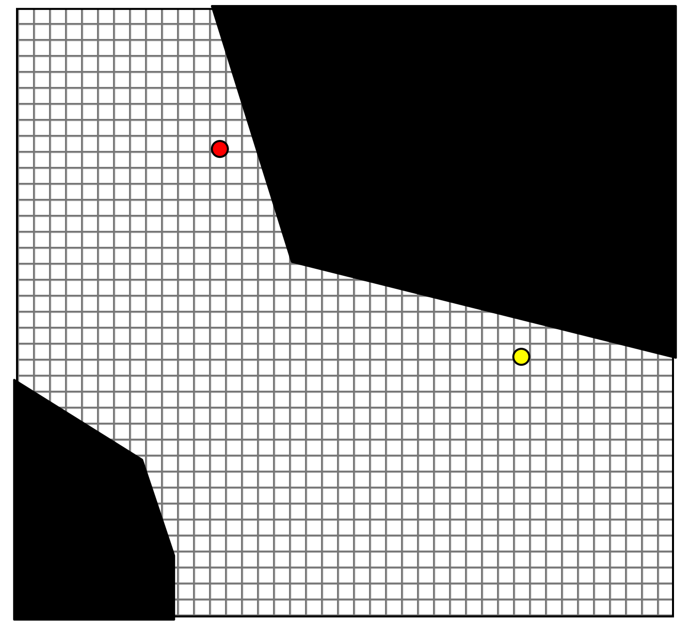
search the graph for a least-cost path from s_{start} to s_{goal}

Grid-based Graphs

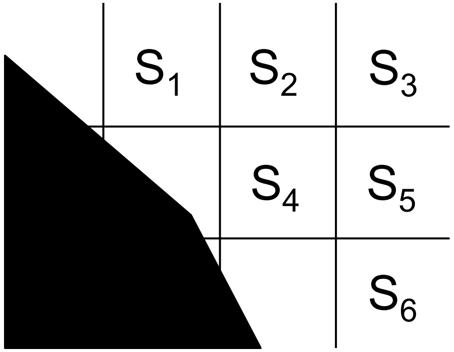
- Approximate Cell Decomposition:
  - construct a graph



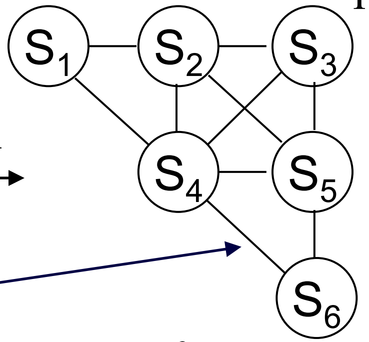
discretize



planning map



convert into a graph

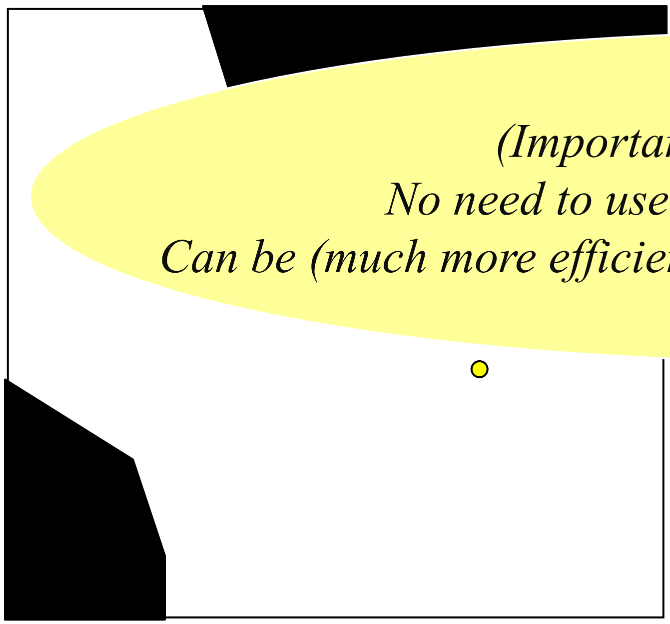


search the graph for a least-cost path from s_{start} to s_{goal}

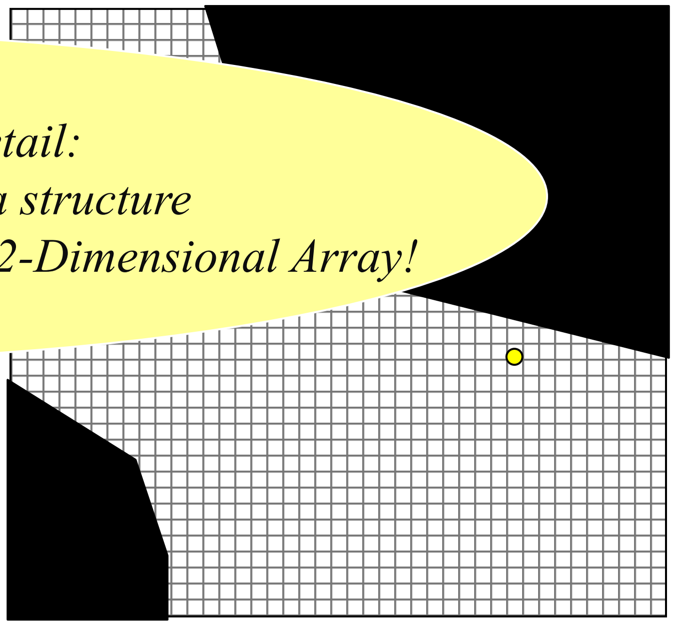
edgecosts can represent ***any*** *cost function*

Grid-based Graphs

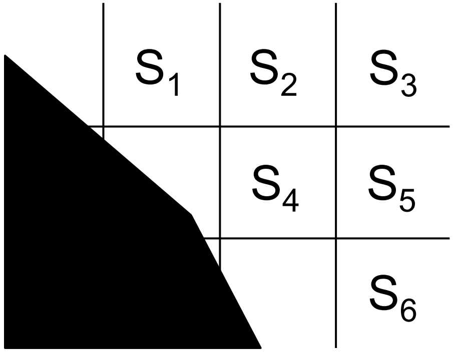
- Approximate Cell Decomposition:
  - construct a graph



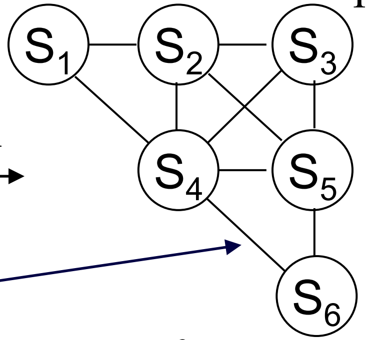
(Important) Implementation Detail:
No need to use an explicit graph data structure
Can be (much more efficiently) represented as a 2-Dimensional Array!



planning map



convert into a graph

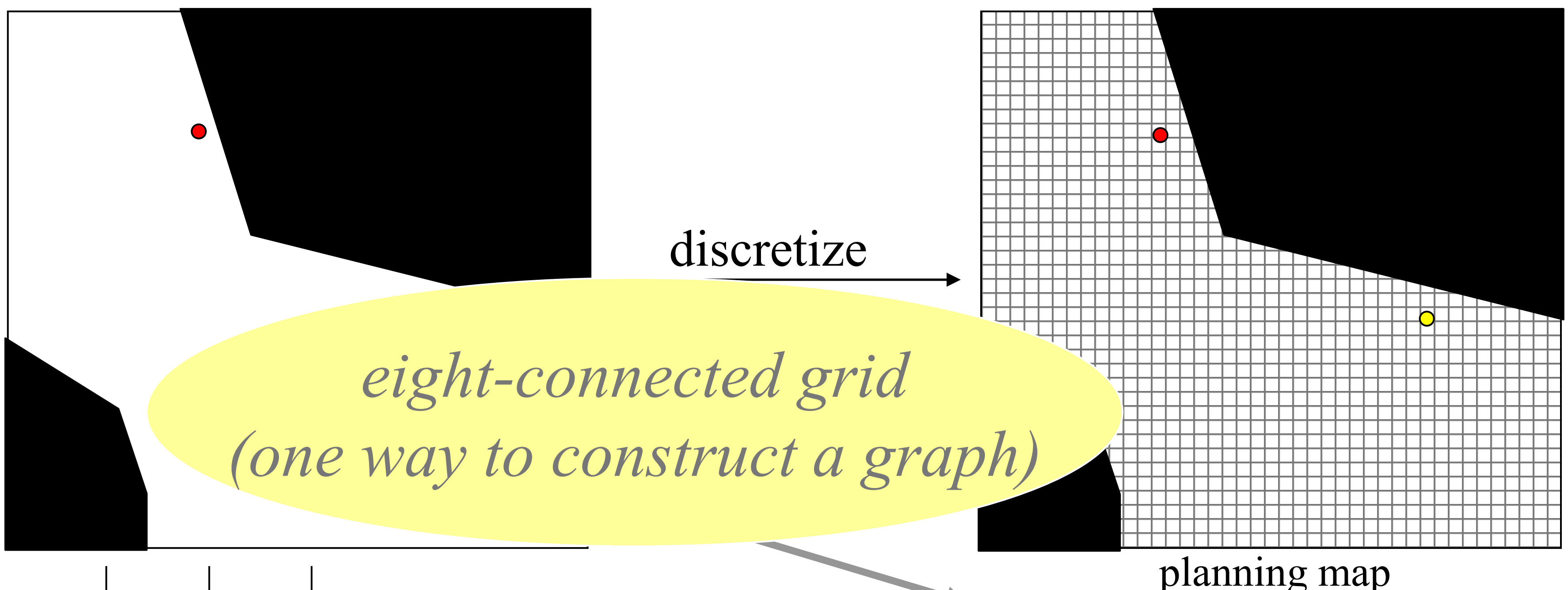


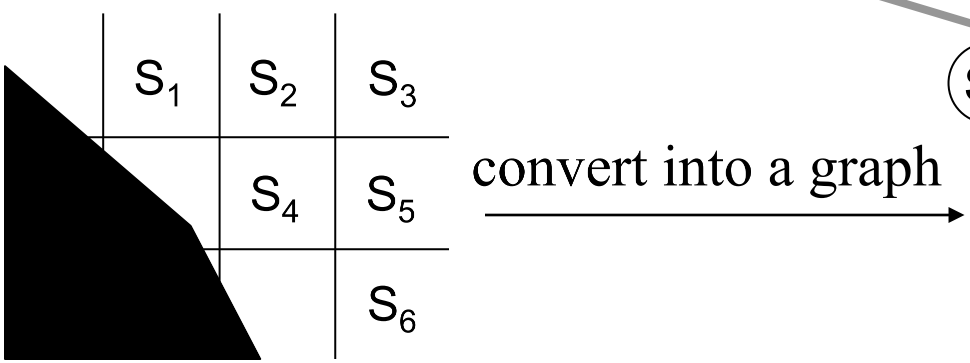
search the graph for a least-cost path from s_{start} to s_{goal}

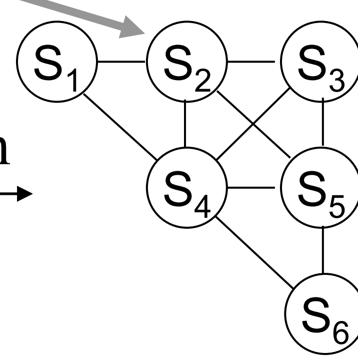
*edgecosts can represent **any** cost function*

Grid-based Graphs

- Approximate Cell Decomposition:
  - construct a graph



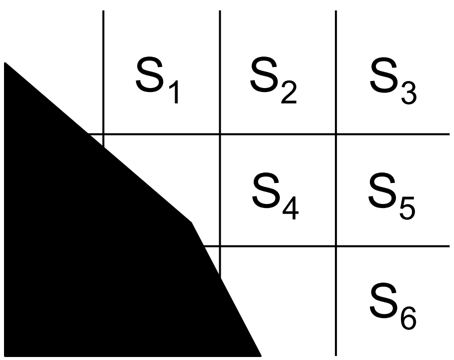




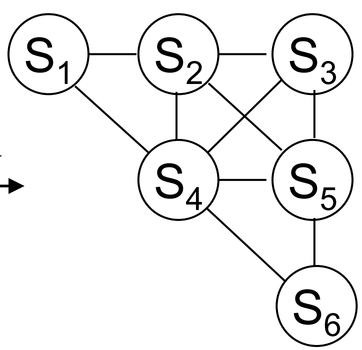
search the graph for a least-cost path from s_{start} to s_{goal}

Grid-based Graphs

- Approximate Cell Decomposition:
  - what to do with partially blocked cells?



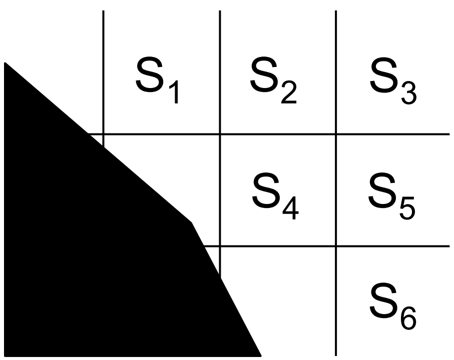
convert into a graph



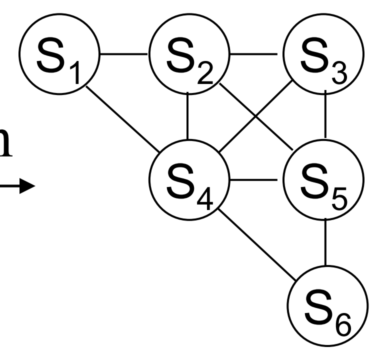
search the graph for a least-cost path from s_{start} to s_{goal}

Grid-based Graphs

- Approximate Cell Decomposition:
  - what to do with partially blocked cells?
  - make it untraversable – incomplete (may not find a path that exists)



convert into a graph

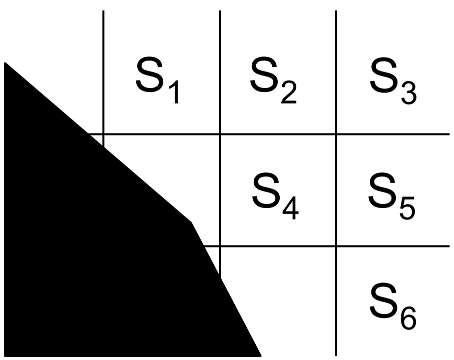


search the graph for a least-cost path from s_{start} to s_{goal}

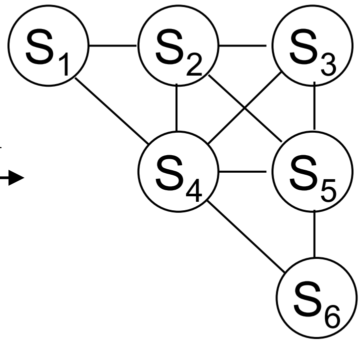
Grid-based Graphs

- Approximate Cell Decomposition:
  - what to do with partially blocked cells?
  - make it traversable – unsound (may return invalid path)

so, what's the solution?



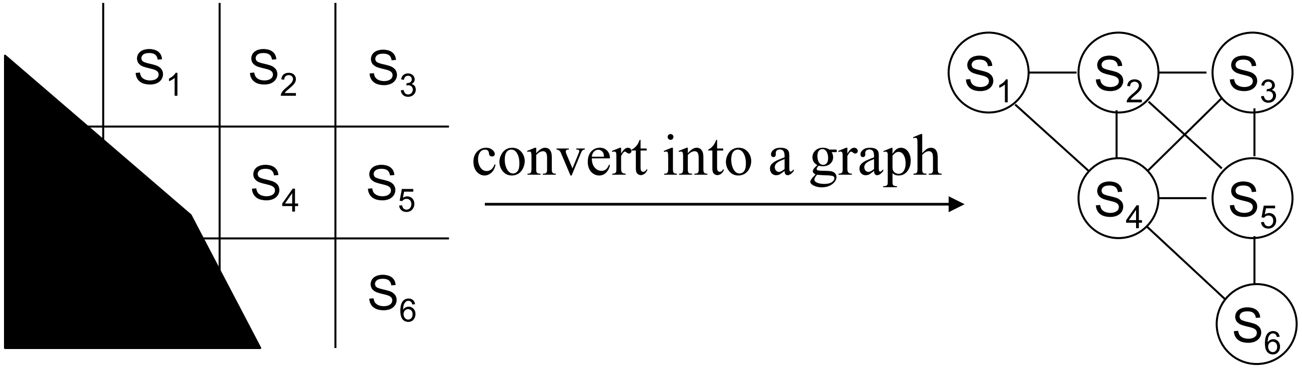
convert into a graph



search the graph for a least-cost path from s_{start} to s_{goal}

Grid-based Graphs

- Approximate Cell Decomposition:
  - solution 1:
    - make the discretization very fine
    - expensive, especially in high-D

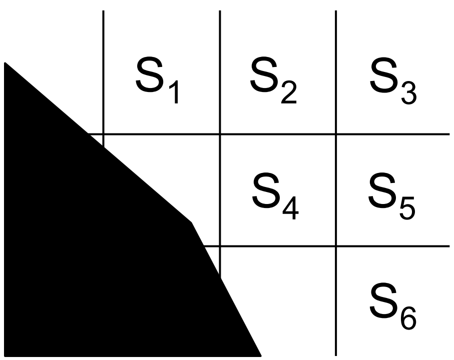


search the graph for a least-cost path from s_{start} to s_{goal}

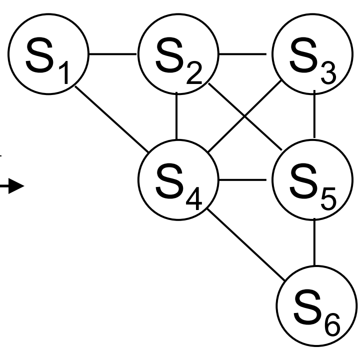
Grid-based Graphs

- Approximate Cell Decomposition:
  - solution 2:
    - make the discretization adaptive
    - various ways possible

Any ideas?



convert into a graph

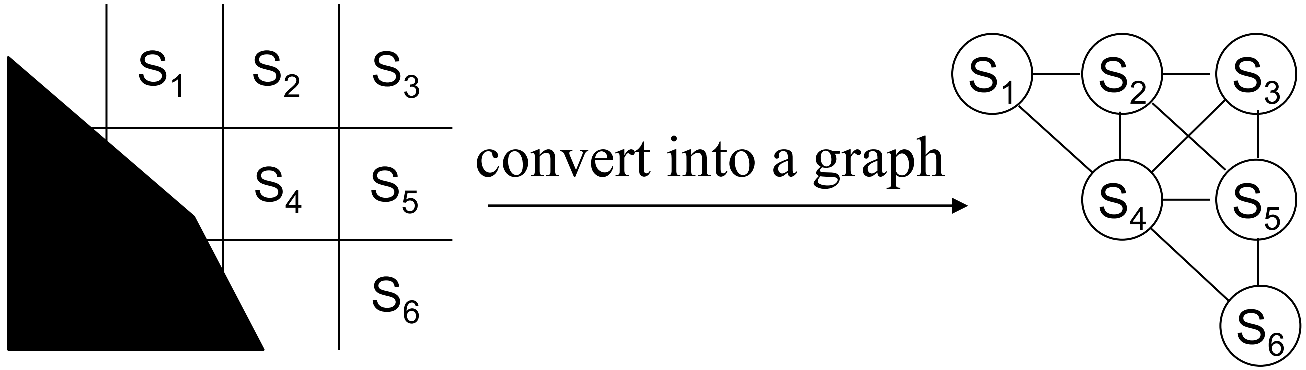


search the graph for a least-cost path from s_{start} to s_{goal}

Grid-based Graphs

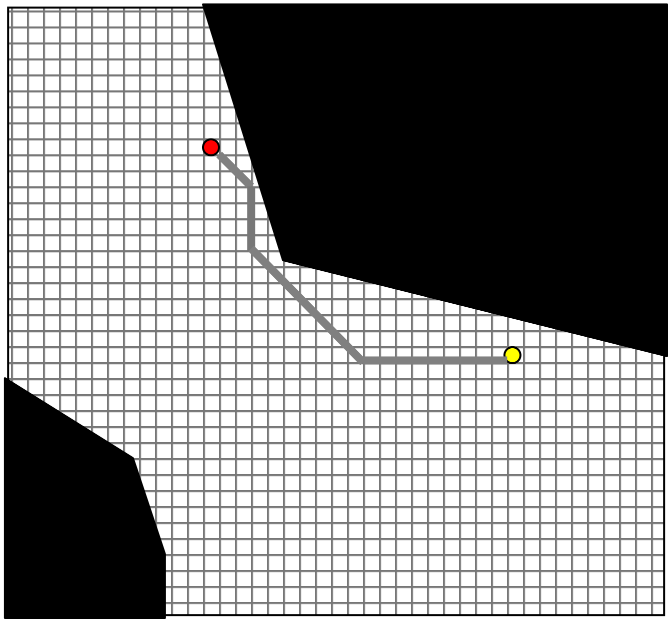
- Graph construction:
  - connect neighbors

***8**-connected grid*



Grid-based Graphs

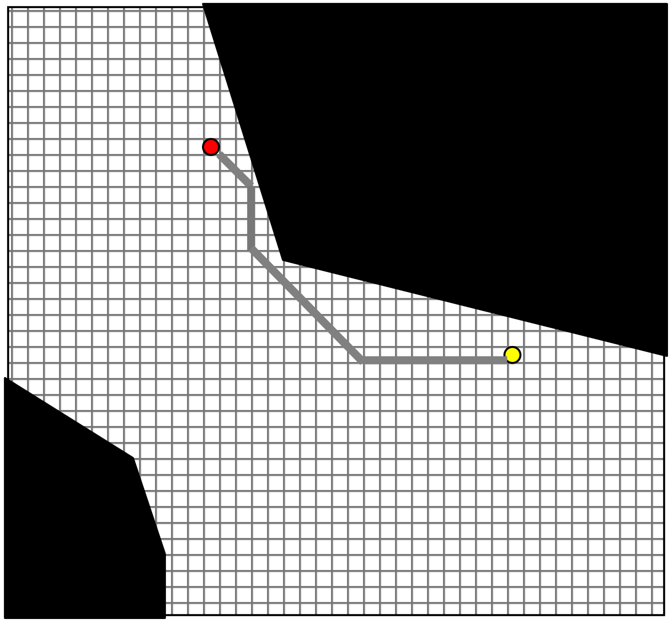
- Graph construction:
  - connect neighbors
  - path is restricted to 45º degrees



Grid-based Graphs

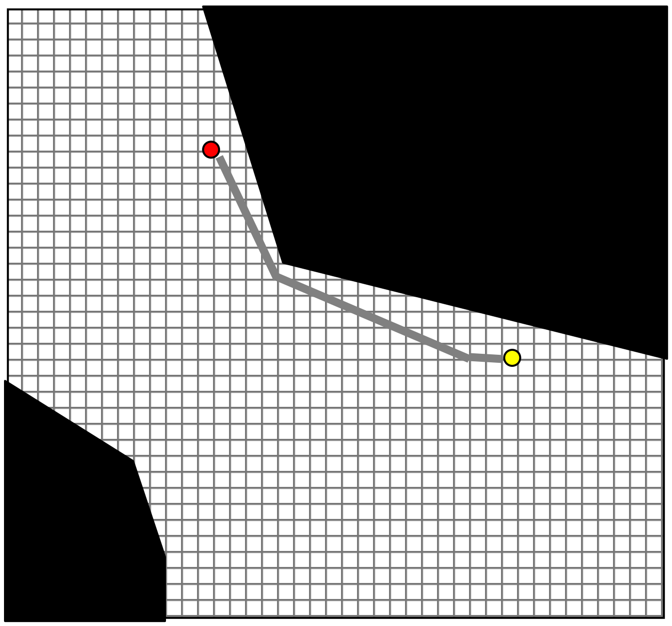
- Graph construction:
  - connect neighbors
  - path is restricted to 45º degrees

Ideas to improve it?

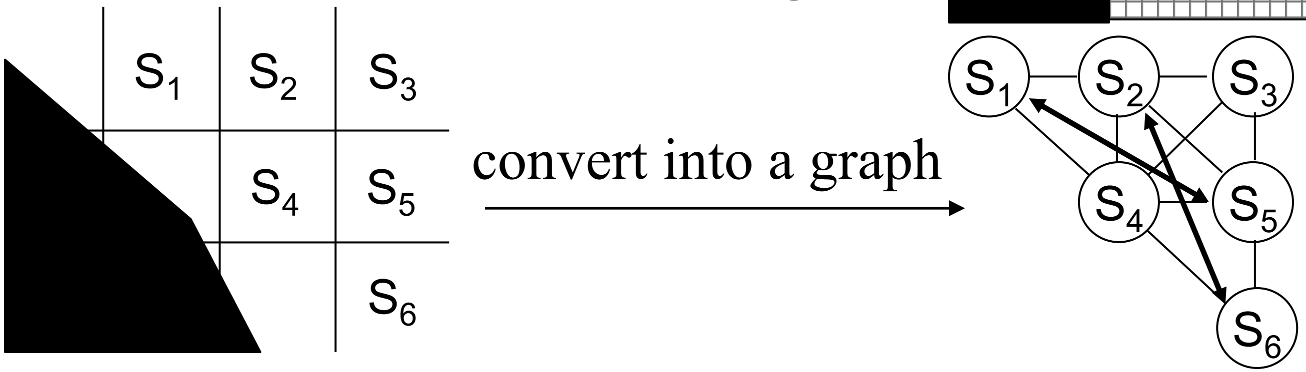


Grid-based Graphs

- Graph construction:
  - connect cells to neighbor of neighbors
  - path is restricted to ? degrees

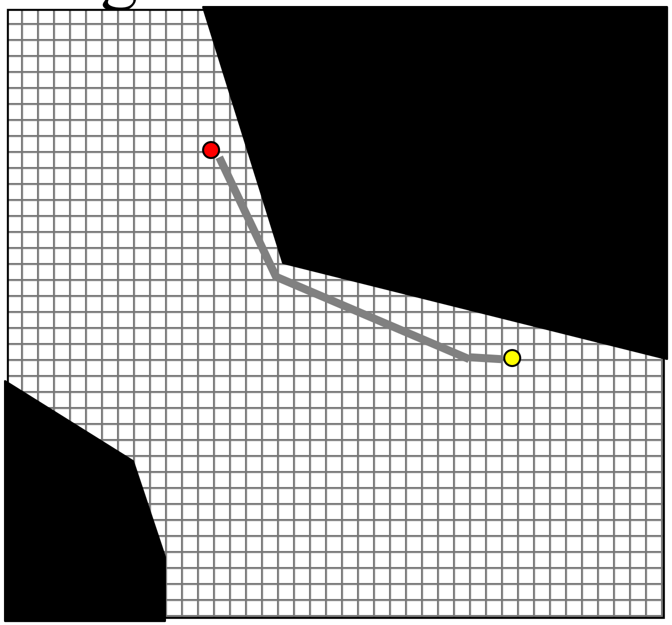


***16**-connected grid*

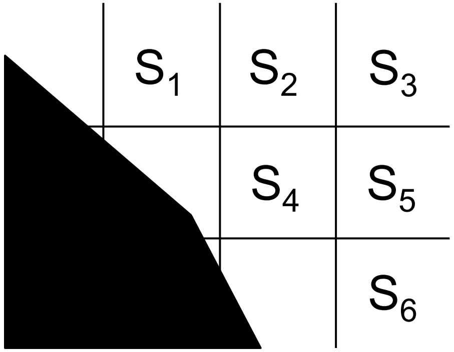


Grid-based Graphs

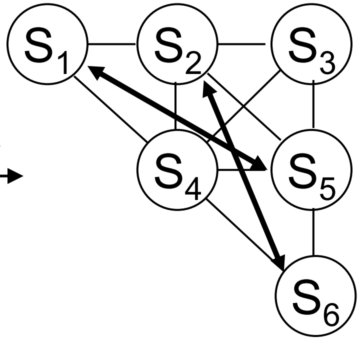
- Graph construction:
  - connect cells to neighbor of neighbors
  - path is restricted to **26.6º/63.4º** degrees



***16**-connected grid*

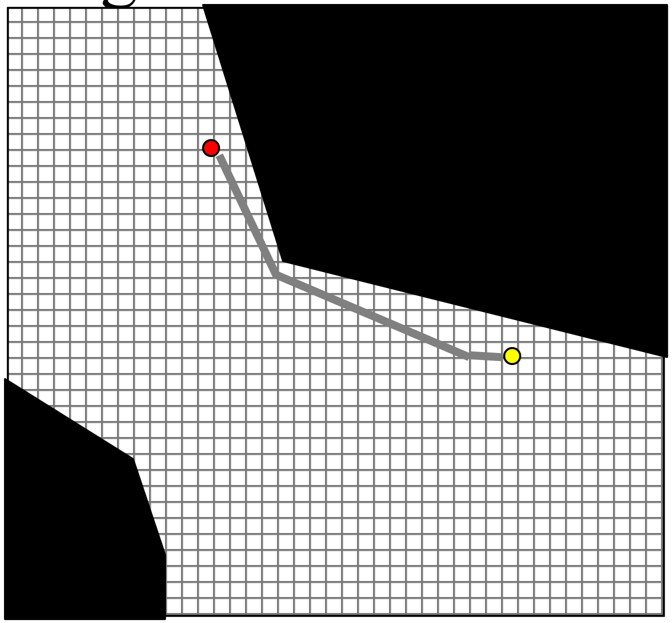


convert into a graph

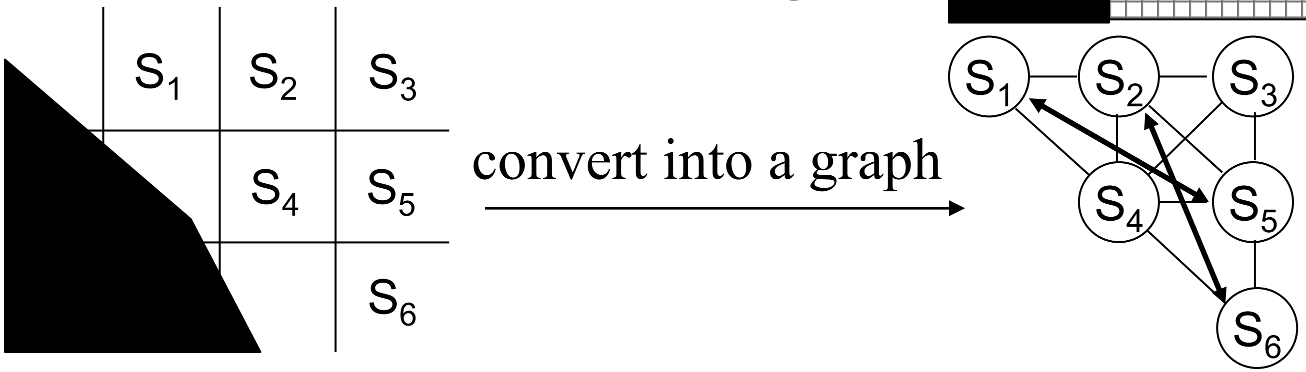


Grid-based Graphs

- Graph construction:
  - connect cells to neighbor of neighbors
  - path is restricted to **26.6º/63.4º** degrees



***16**-connected grid*



Cell Decomposition-based Graphs

- Grid-based graph
  - advantages:
    - very simple to implement (super popular)
    - can represent any dimensional space
    - works well with obstacles represented as set of points
    - works with any cost function
  - disadvantages:
    - size does depend on the size of the environment
    - expensive to maintain/compute grids of dimensions > 3

Cell Decomposition-based Graphs

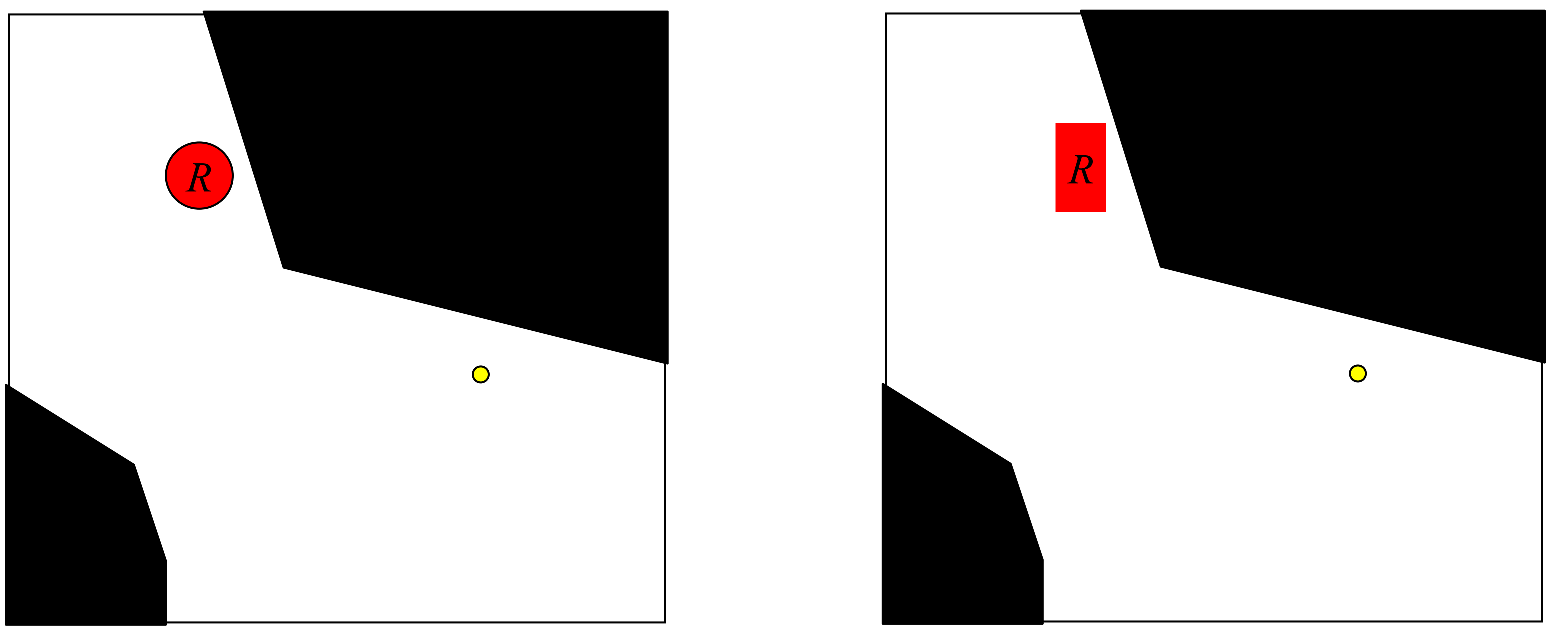
- Grid-based graph
  - advantages:
    - very simple to implement (super popular)
    - can represent any dimensional space
    - works well with obstacles represented as set of points
    - works with any cost function
  - disadvantages:
    - size does depend on the size of the environment
    - expensive to maintain/compute grids of dimensions > 3

More on this in a later class on
Implicit Graph representations for high-dimensional planning problems

2D Planning for Omnidirectional **Non-Circular Non-point** Robot

Planning for omnidirectional point robot:

What is $M^R = \langle x,y \rangle$
What is $M^W = \langle obstacle/free\ space \rangle$
What is $s^R_{current} = \langle x_{current}, y_{current} \rangle$
What is $s^W_{current} = constant$
What is $C = Euclidean\ Distance$
What is $G = \langle x_{goal}, y_{goal} \rangle$

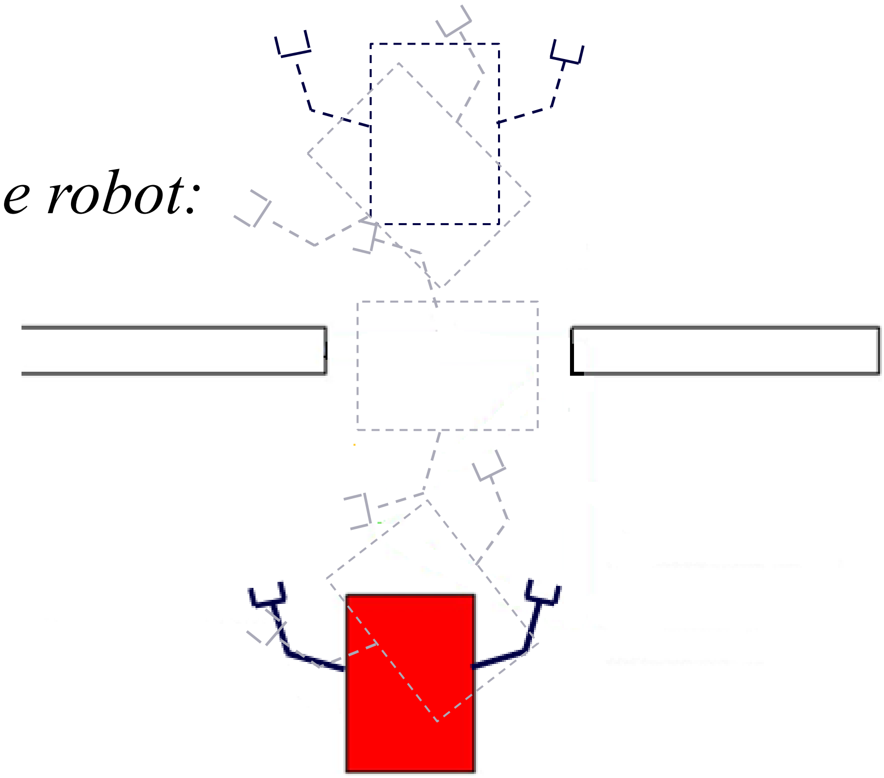


Configuration Space

- Configuration is legal if it does not intersect any obstacles and is valid

- Configuration Space is the set of legal configurations

Legal configurations for the base of the robot:

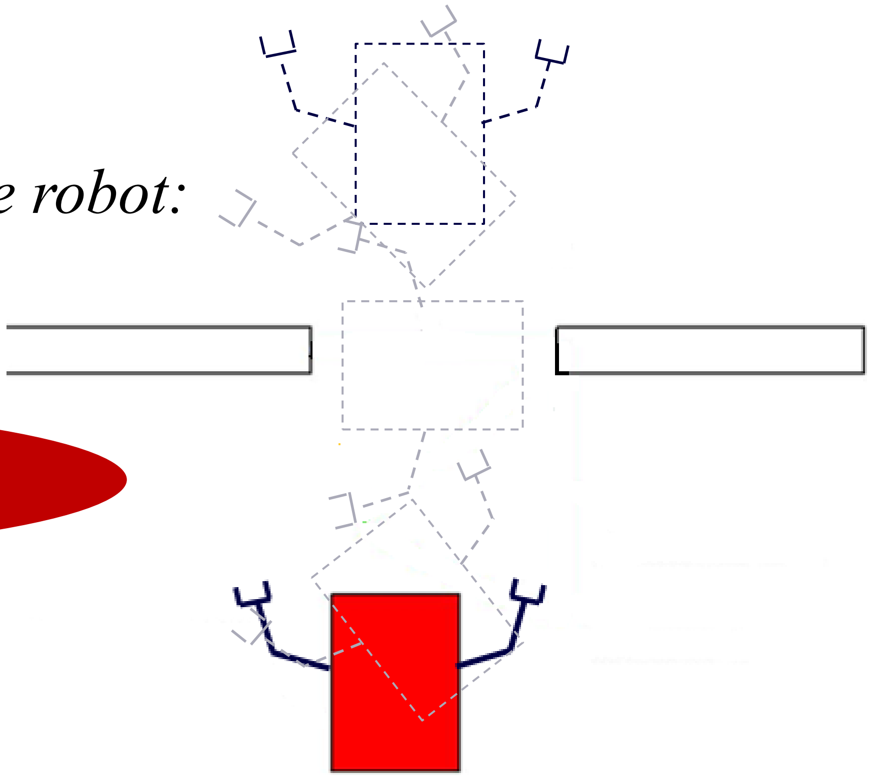


Configuration Space

- Configuration is legal if it does not intersect any obstacles and is valid

- Configuration Space is the set of legal configurations

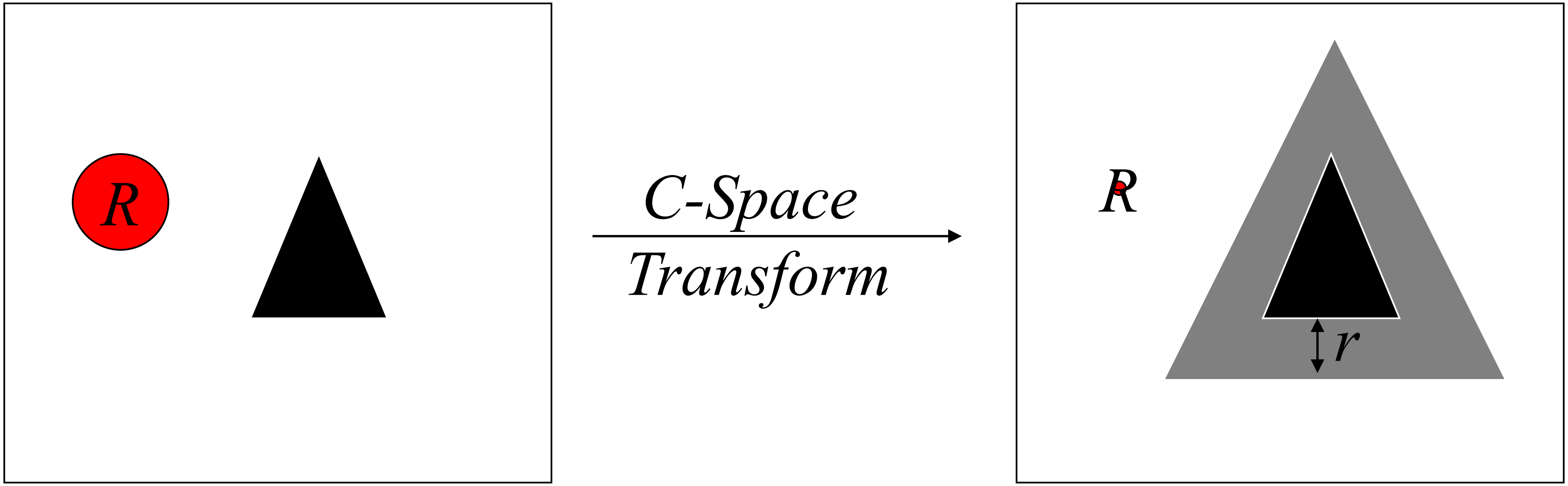
Legal configurations for the base of the robot:



What is the dimensionality of this configuration space?

C-Space Transform

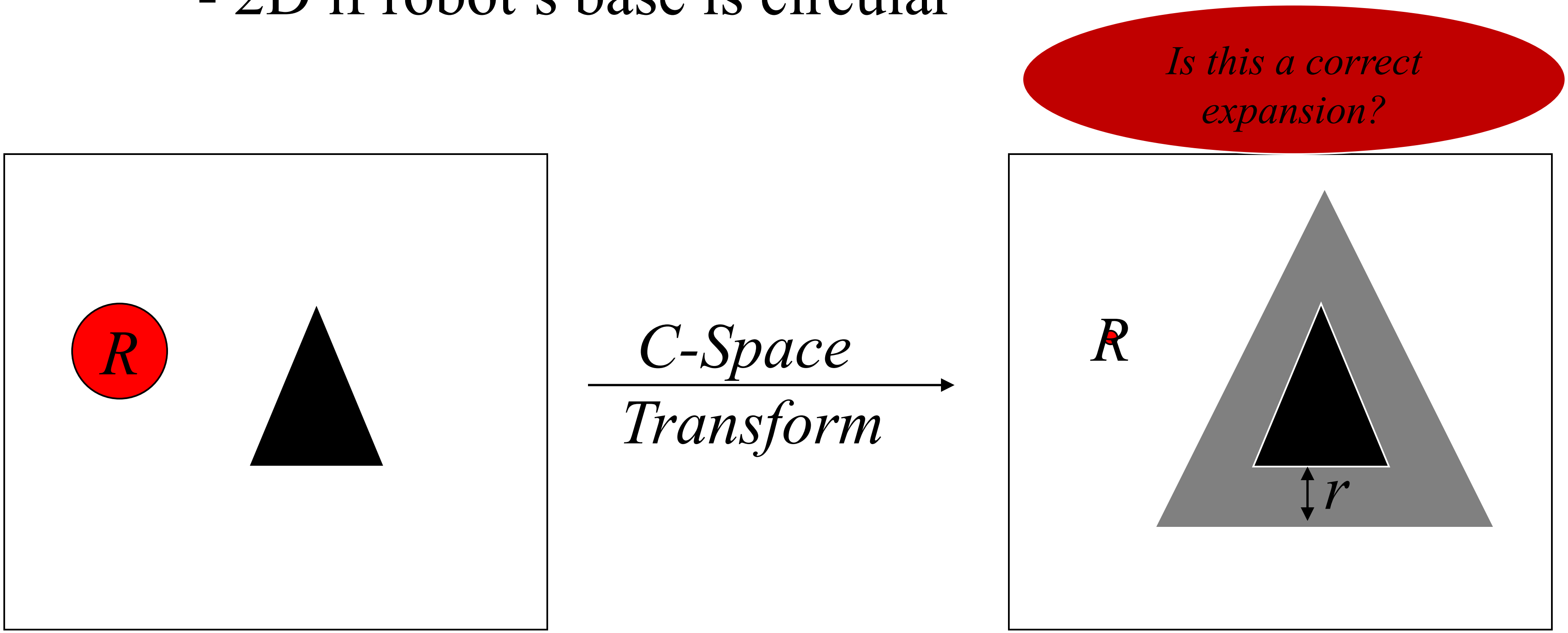
- Configuration space for a robot base in 2D world is:
  - 2D if robot's base is circular



- expand all obstacles by radius r of the robot's base
- graph construction can then be done assuming point robot

C-Space Transform

- Configuration space for a robot base in 2D world is:
  - 2D if robot's base is circular

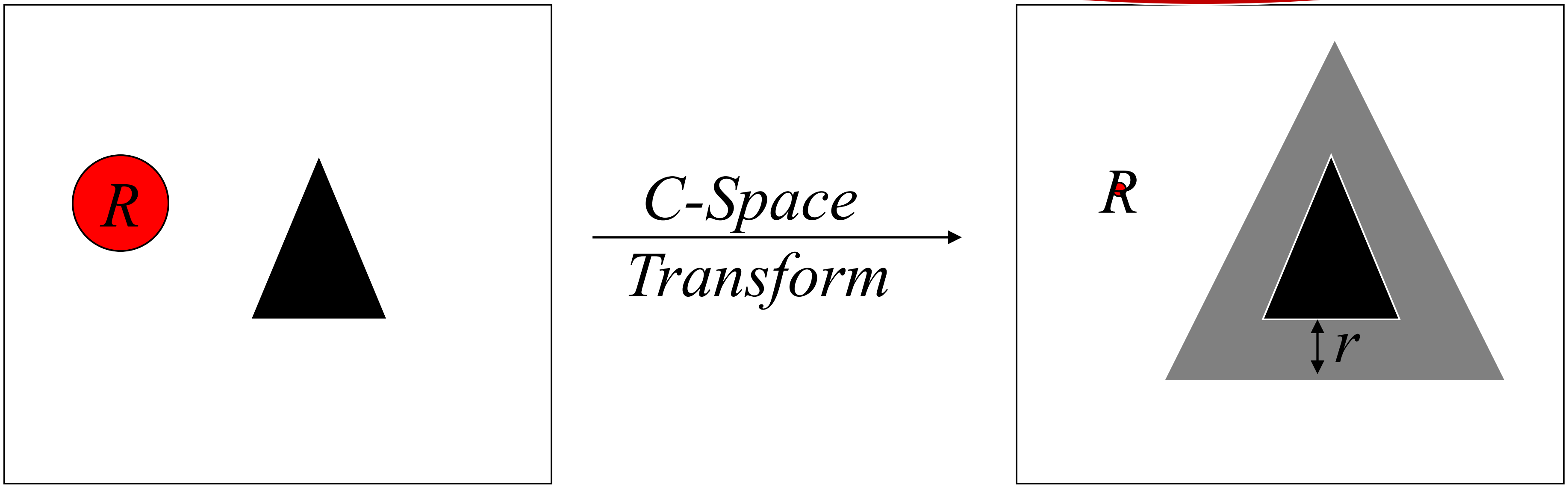


- expand all obstacles by radius r of the robot's base
- graph construction can then be done assuming point robot

C-Space Transform

- Configuration space for a robot base in 2D world is:
  - 2D if robot's base is circular

How to perform expansion of obstacles?



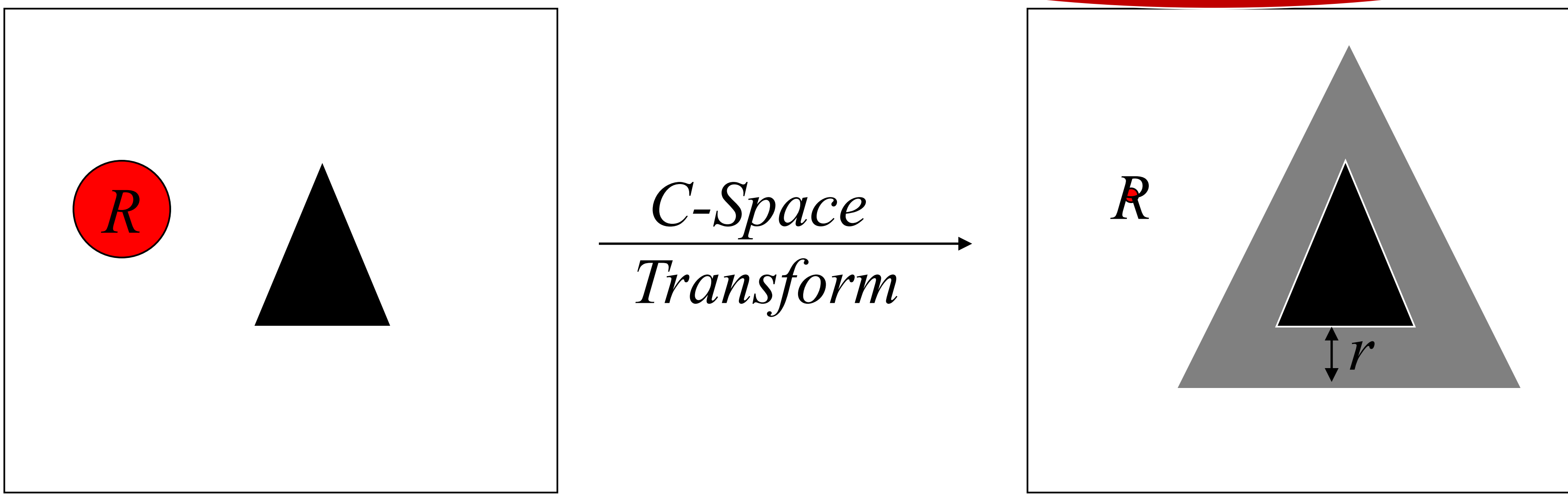
- expand all obstacles by radius r of the robot's base
- graph construction can then be done assuming point robot

C-Space Transform

- Configuration space for a robot ba[...]
  - 2D if robot's base is circula[...]

O(n) methods exist to compute distance transforms efficiently

How to perform expansion of obstacles?

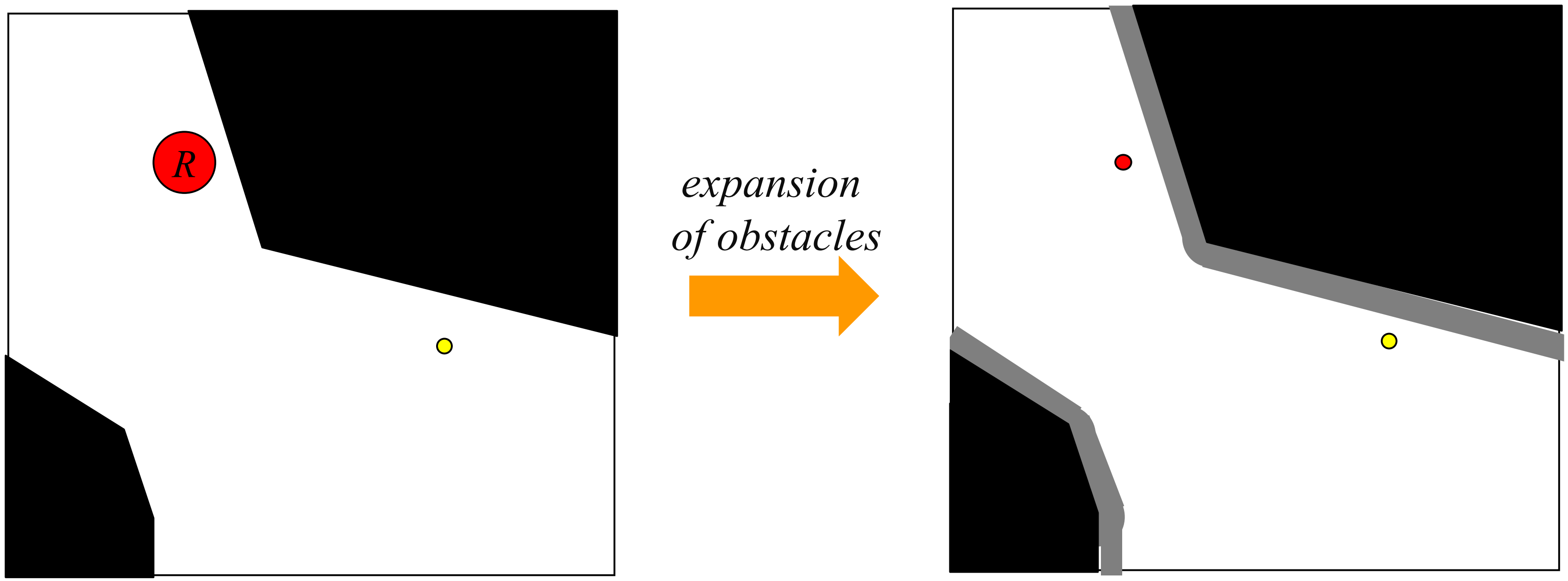


- expand all obstacles by radius r of the robot's base
- graph construction can then be done assuming point robot

2D Planning for Omnidirectional **Non-Circular Non-point** Robot

Planning for omnidirectional circular robot:

What is $M^R = <x,y>$
What is $M^W = <obstacle/free\ space>$
What is $s^R_{current} = <x_{current}, y_{current}>$
What is $s^W_{current} = constant$
What is C = *Euclidean Distance*
What is $G = <x_{goal}, y_{goal}>$

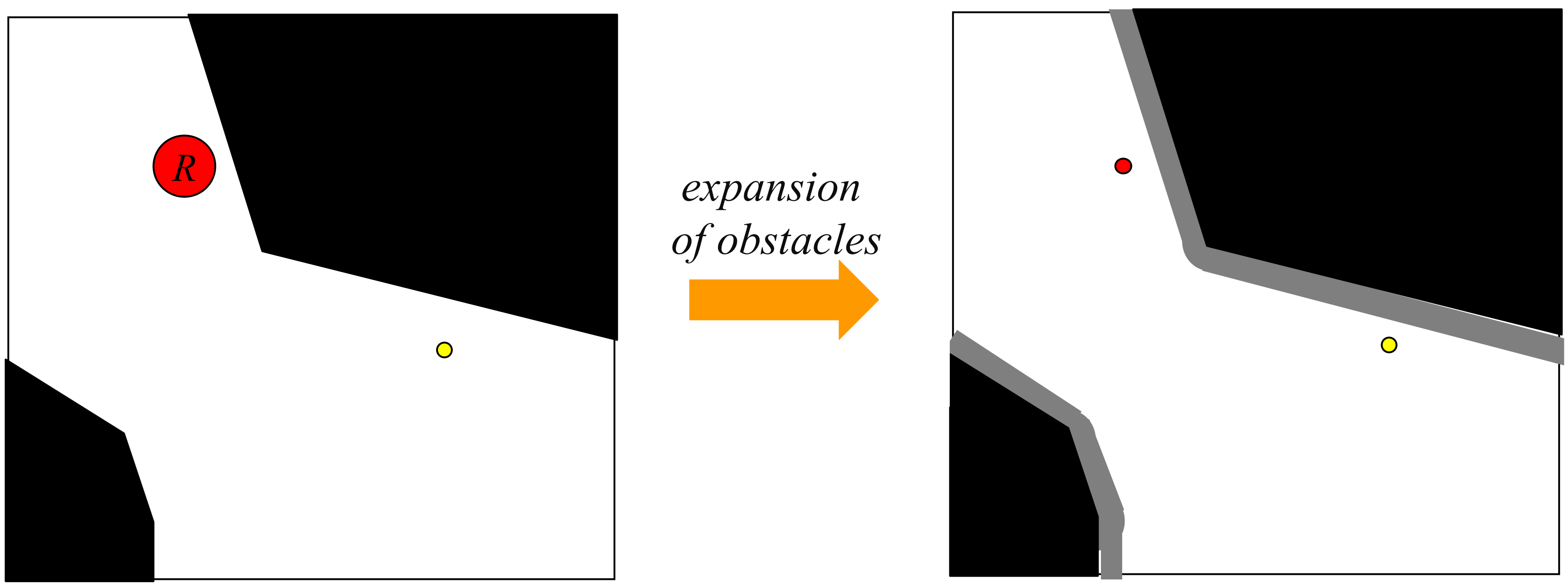


2D Planning for Omnidirectional **Non-Circular Non-point** Robot

Planning for omnidirectional circular robot:

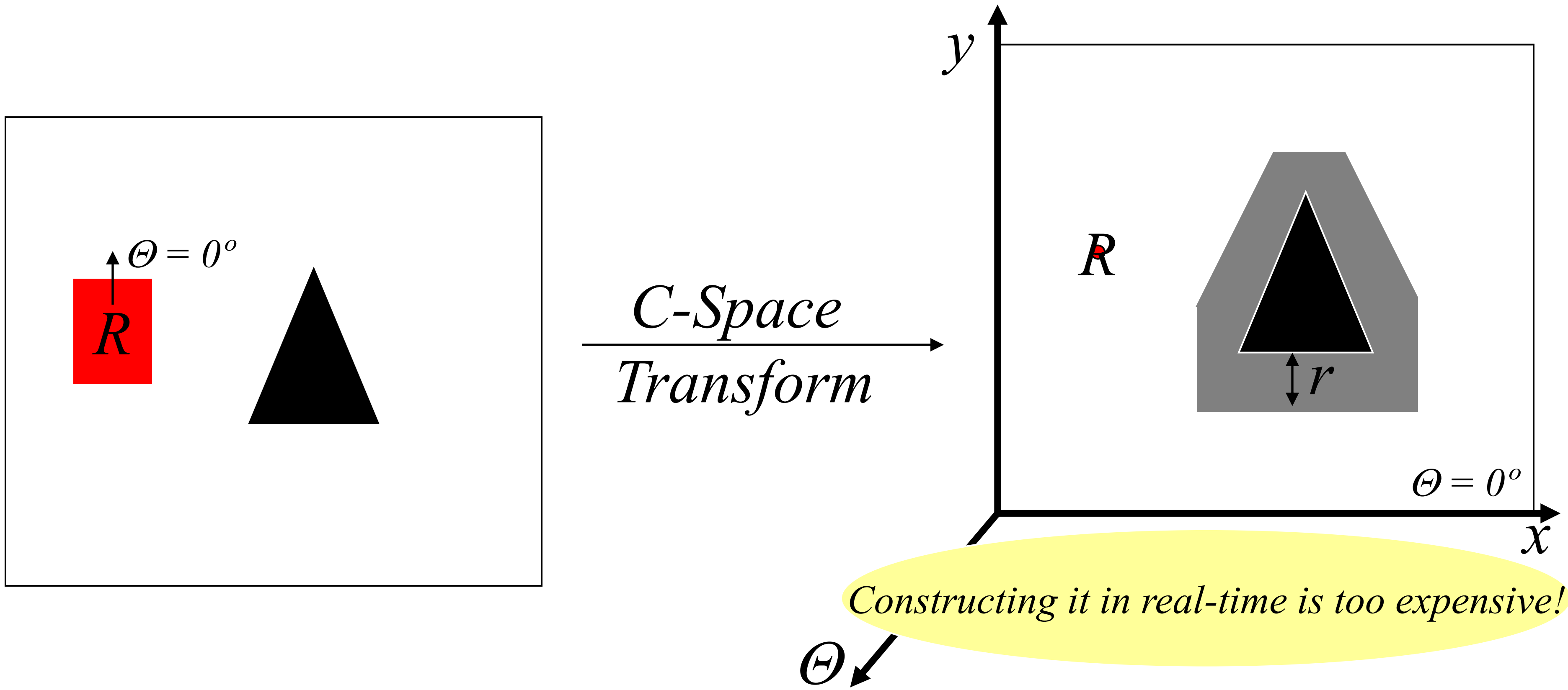
What is $M^R = \langle x,y \rangle$
What is $M^W = \langle \text{obstacle/free space} \rangle$
What is $s^R_{current} = \langle x_{current}, y_{current} \rangle$
What is $s^W_{current}$ = *constant*
What is C = *Euclidean Distance*
What is $G = \langle x_{goal}, y_{goal} \rangle$

We can now construct a graph using previously discussed methods (grids, Voronoi graphs, Visibility graphs)



C-Space Transform

- Configuration space for a robot base in 2D world is:
  - 3D if robot's base is non-circular



Planning as Graph Search Problem

1. Construct a graph representing the planning problem

2. Search the graph for a (hopefully, close-to-optimal) path

The two steps above are often interleaved

Planning as Graph Search Problem

1. Construct a graph representing the planning problem

2. Search the graph for a (hopefully, close-to-optimal) path

The two steps above are often interleaved

Interleaving Search and Graph Construction

Graph Search using an **Explicit Graph** (allocated prior to the search itself):

1. *Create the graph G = {V, E} in-memory*

2. *Search the graph*

Using Explicit Graphs
is typical for low-D (i.e., 2D) problems in Robotics
(with the exception of PRMs, covered in a later lecture)

Interleaving Search and Graph Construction

Graph Search using an **Implicit Graph** (allocated as needed by the search):

1. *Instantiate Start state*

2. *Start searching with the Start state using functions*

   a) *Succs = GetSuccessors (State s, Action)*
   b) *ComputeEdgeCost (State s, Action a, State s')*

   and allocating memory for the generated states

Using Implicit Graphs
is critical for most (>2D) problems
in Robotics

Interleaving Search and Graph Construction

- **Board example** for deciding whether to use an Explicit graph or Implicit graph

- Planning for (x,y,Θ) for
  - 20 by 20 m environment discretized into 25 cm cells with 8 heading Θ values

Is it feasible to use Explicit Graph (memory and pre-computation time reqs)?

Interleaving Search and Graph Construction

- **Board example** for deciding whether to use an Explicit graph or Implicit graph

- Planning for *(x,y,Θ)* for
  - 200 by 200 m environment discretized into 25 cm cells with 16 heading Θ values for a real vehicle

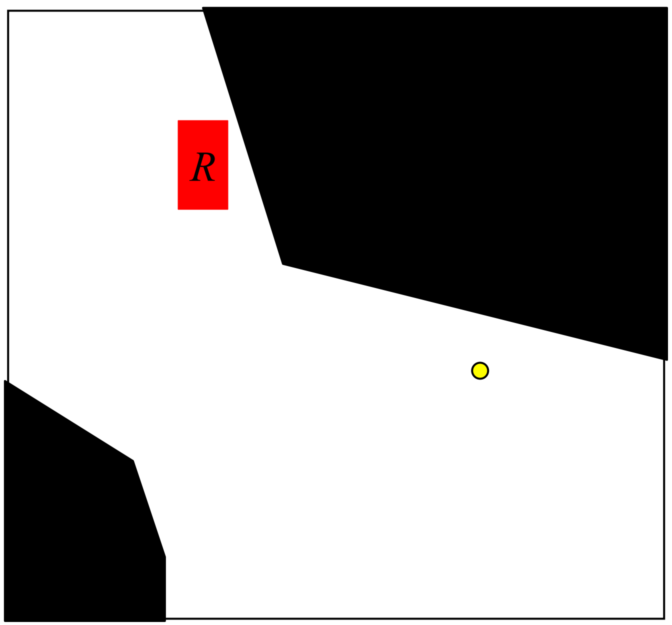
Is it feasible to use Explicit Graph (memory and pre-computation time reqs)?

2D Planning for Omnidirectional **Non-Circular Non-point** Robot

Planning for omnidirectional non-circular robot:

What is $M^R = \langle x, y, \Theta \rangle$
What is $M^W = \langle$*obstacle/free space*$\rangle$
What is $s^R_{current} = \langle x_{current}, y_{current}, \Theta_{current} \rangle$
What is $s^W_{current}$ = *constant*
What is C = Euclidean Distance
What is $G = \langle x_{goal}, y_{goal}, \Theta_{goal} \rangle$

Interleave
Graph Construction and Graph Search steps!



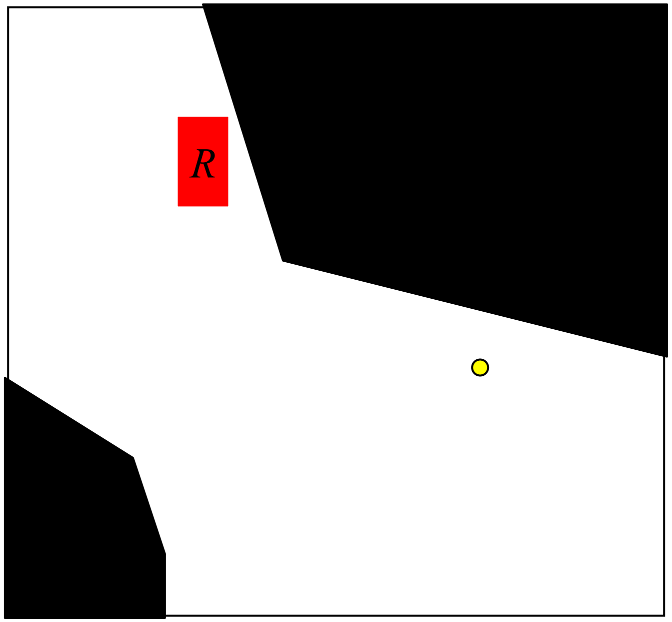
Construct a 3D grid (x,y,Θ) assuming point robot (i.e., a cell (x,y,Θ) is free whenever its (x,y) is free) and compute the ***actual*** *validity of only those cells that get computed by the graph search*

2D Planning for Omnidirectional **Non-Circular Non-point** Robot

Planning for omnidirectional non-circular robot:

What is $M^R = <x,y,\Theta>$
What is $M^W = <obstacle/free\ space>$
What is $s^R_{current} = <x_{current}, y_{current}, \Theta_{current}>$
What is $s^W_{current} = constant$
What is C = *Euclidean Distance*
What is $G = <x_{goal}, y_{goal}, \Theta_{goal}>$

Interleave
Graph Construction and Graph Search steps!



*Construct a 3D grid (x,y,Θ) assuming point robot (i.e., a cell (x,y,Θ) is free whenever its (x,y) is free) and compute the **actual** validity of only those cells that get computed by the graph search*

How to compute the actual validity of cell (x,y,Θ)?

2D Planning for Omnidirectional **Non-Circular Non-point** Robot

Planning for omnidirectional non-circular robot:

What is $M^R = <x,y,\Theta>$

What is $M^W = <obstacle/free\ space>$

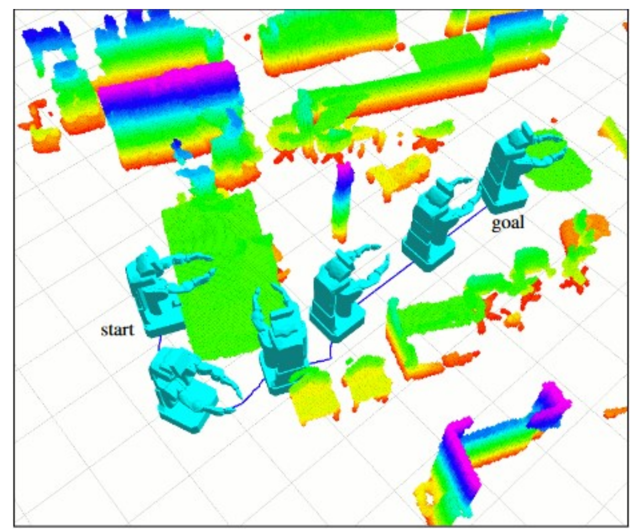
What is $s^R_{current} = <x_{current}, y_{current}, \Theta_{current}>$

What is $s^W_{current} = constant$

What is $C = Euclidean\ Distance$

What is $G = <x_{goal}, y_{goal}, \Theta_{goal}>$

Interleave
Graph Construction and Graph Search steps!



What You Should Know…

- What visibility graphs are
- What Voronoi diagram-based graphs are
- X-connected N-dimensional grids
- Configuration Space, C-Space Transform
- Explicit vs. Implicit graphs and pros/cons of each