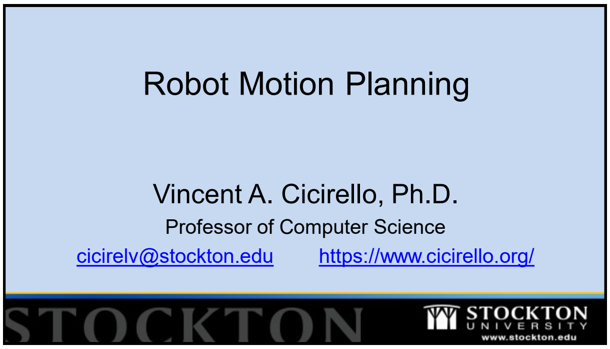
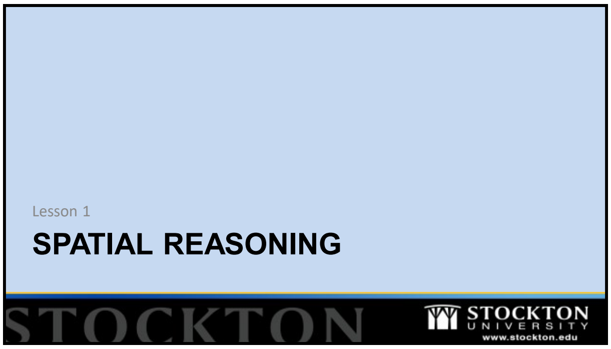


# Robot Motion Planning

Vincent A. Cicirello, Ph.D.
Professor of Computer Science
cicirelv@stockton.edu https://www.cicirello.org/

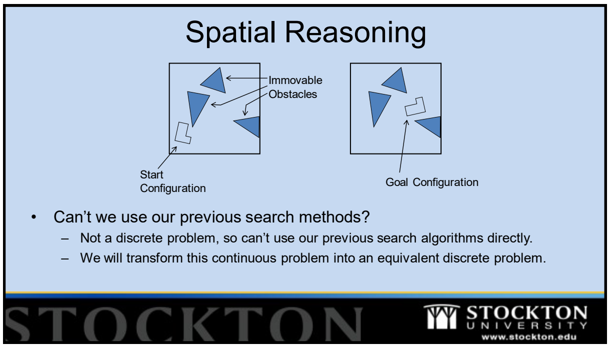
1



Lesson 1

## SPATIAL REASONING

2



## Spatial Reasoning

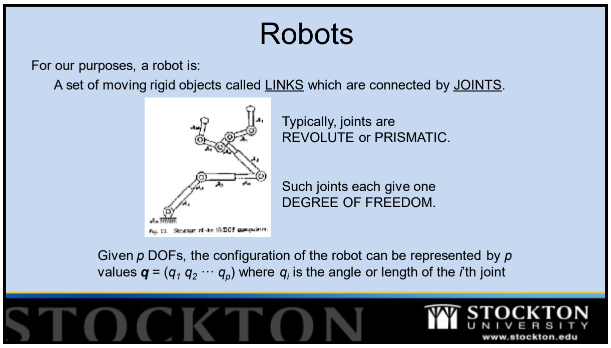
Immovable Obstacles

Start Configuration

Goal Configuration

- Can't we use our previous search methods?
  - Not a discrete problem, so can't use our previous search algorithms directly.
  - We will transform this continuous problem into an equivalent discrete problem.

3



## Robots

For our purposes, a robot is:

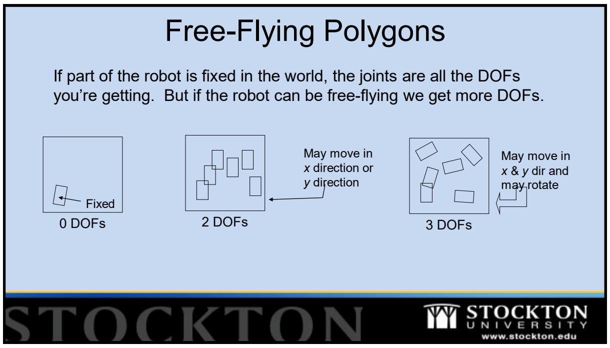
A set of moving rigid objects called LINKS which are connected by JOINTS.

Typically, joints are REVOLUTE or PRISMATIC.

Such joints each give one DEGREE OF FREEDOM.

Given *p* DOFs, the configuration of the robot can be represented by *p* values $\boldsymbol{q} = (q_1\ q_2 \cdots q_p)$ where $q_i$ is the angle or length of the *i*'th joint

4



## Free-Flying Polygons

If part of the robot is fixed in the world, the joints are all the DOFs you're getting. But if the robot can be free-flying we get more DOFs.

Fixed

0 DOFs

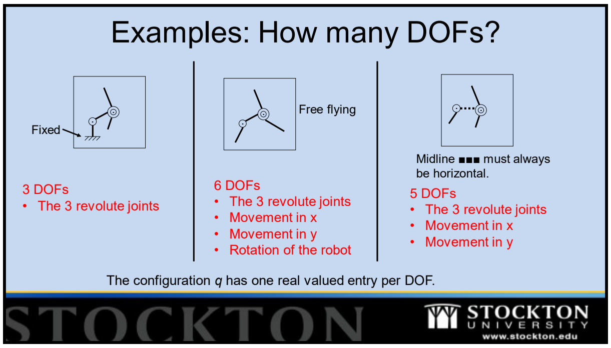
May move in *x* direction or *y* direction

2 DOFs

May move in *x* & *y* dir and may rotate

3 DOFs

5



## Examples: How many DOFs?

Fixed

3 DOFs
- The 3 revolute joints

Free flying

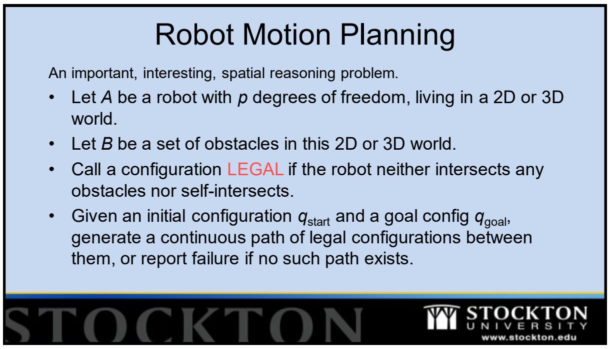
6 DOFs
- The 3 revolute joints
- Movement in x
- Movement in y
- Rotation of the robot

Midline ■■■ must always be horizontal.

5 DOFs
- The 3 revolute joints
- Movement in x
- Movement in y

The configuration *q* has one real valued entry per DOF.

6

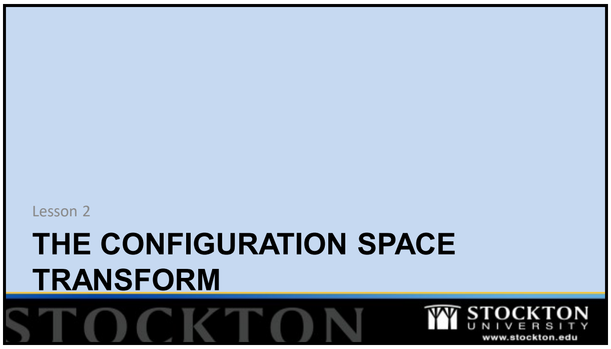


## Robot Motion Planning

An important, interesting, spatial reasoning problem.

- Let *A* be a robot with *p* degrees of freedom, living in a 2D or 3D world.
- Let *B* be a set of obstacles in this 2D or 3D world.
- Call a configuration LEGAL if the robot neither intersects any obstacles nor self-intersects.
- Given an initial configuration $q_{start}$ and a goal config $q_{goal}$, generate a continuous path of legal configurations between them, or report failure if no such path exists.

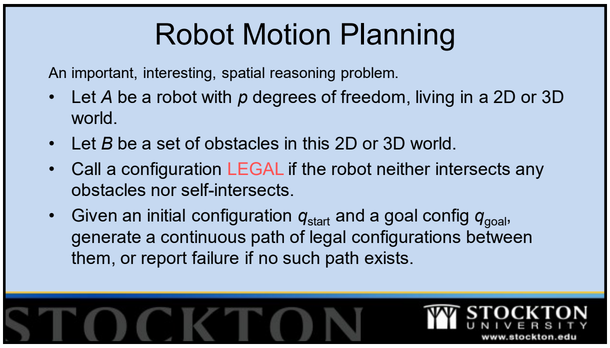
7



Lesson 2

## THE CONFIGURATION SPACE TRANSFORM

8

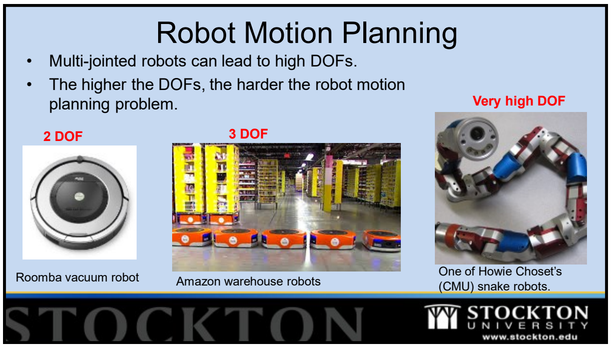


## Robot Motion Planning

An important, interesting, spatial reasoning problem.

- Let *A* be a robot with *p* degrees of freedom, living in a 2D or 3D world.
- Let *B* be a set of obstacles in this 2D or 3D world.
- Call a configuration LEGAL if the robot neither intersects any obstacles nor self-intersects.
- Given an initial configuration $q_{start}$ and a goal config $q_{goal}$, generate a continuous path of legal configurations between them, or report failure if no such path exists.

9



## Robot Motion Planning

- Multi-jointed robots can lead to high DOFs.
- The higher the DOFs, the harder the robot motion planning problem.

2 DOF

Roomba vacuum robot

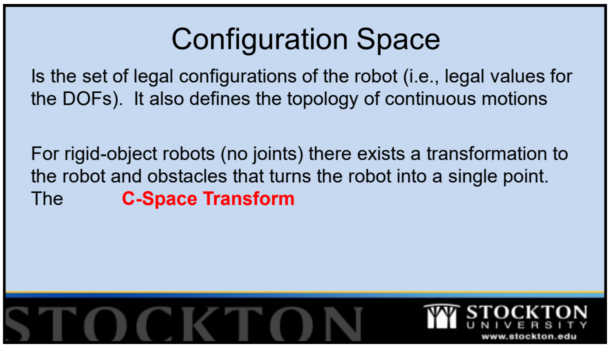
3 DOF

Amazon warehouse robots

Very high DOF

One of Howie Choset's (CMU) snake robots.

10

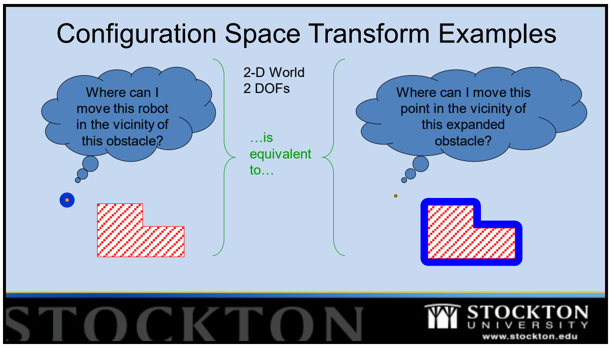


## Configuration Space

Is the set of legal configurations of the robot (i.e., legal values for the DOFs). It also defines the topology of continuous motions

For rigid-object robots (no joints) there exists a transformation to the robot and obstacles that turns the robot into a single point. The **C-Space Transform**

11



## Configuration Space Transform Examples

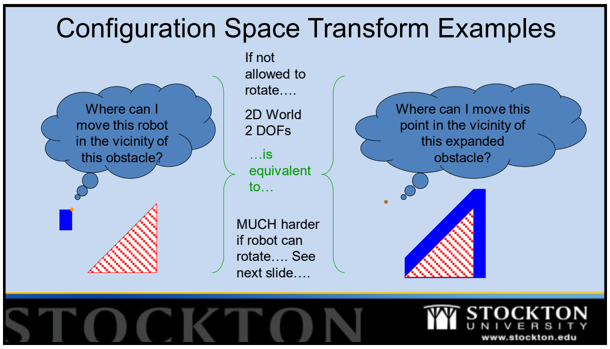
Where can I move this robot in the vicinity of this obstacle?

2-D World
2 DOFs

…is equivalent to…

Where can I move this point in the vicinity of this expanded obstacle?

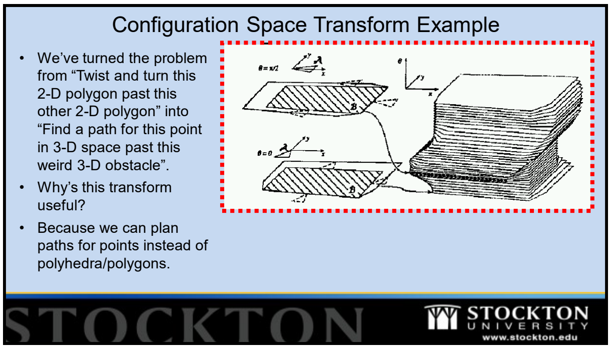
12



## Configuration Space Transform Examples

Where can I move this robot in the vicinity of this obstacle?
If not allowed to rotate….
2D World
2 DOFs
…is equivalent to…
MUCH harder if robot can rotate…. See next slide….
Where can I move this point in the vicinity of this expanded obstacle?

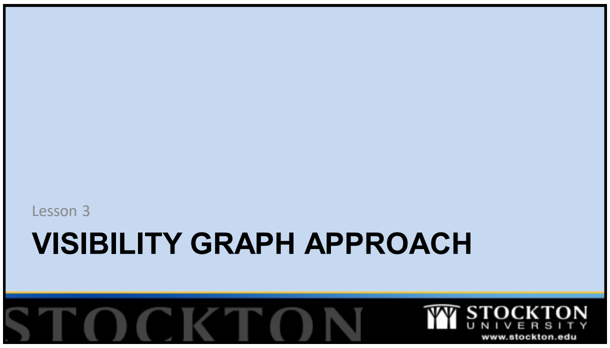
13



## Configuration Space Transform Example

- We've turned the problem from "Twist and turn this 2-D polygon past this other 2-D polygon" into "Find a path for this point in 3-D space past this weird 3-D obstacle".
- Why's this transform useful?
- Because we can plan paths for points instead of polyhedra/polygons.

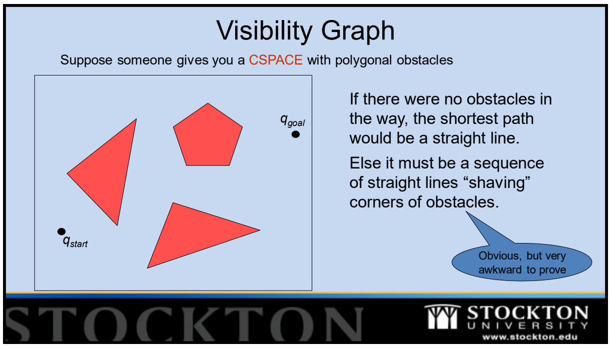
14



Lesson 3

## VISIBILITY GRAPH APPROACH

15



## Visibility Graph

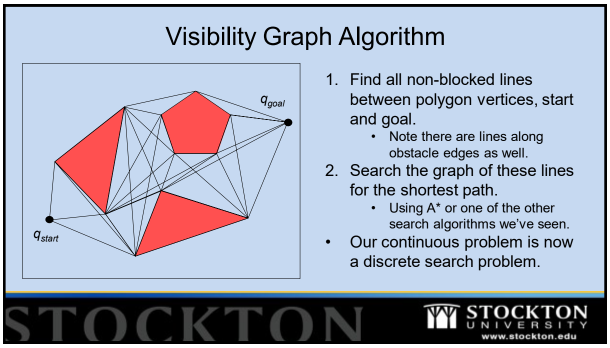
Suppose someone gives you a CSPACE with polygonal obstacles

If there were no obstacles in the way, the shortest path would be a straight line.

Else it must be a sequence of straight lines "shaving" corners of obstacles.

Obvious, but very awkward to prove

16

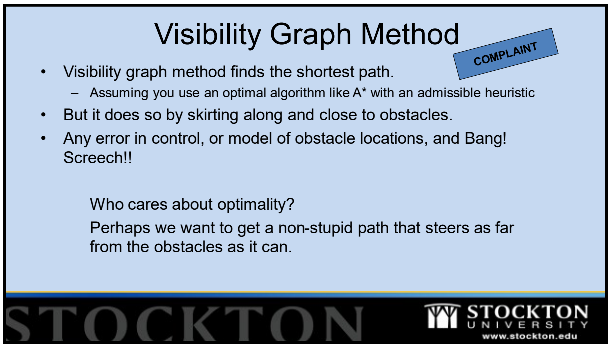


## Visibility Graph Algorithm

1. Find all non-blocked lines between polygon vertices, start and goal.
    - Note there are lines along obstacle edges as well.
2. Search the graph of these lines for the shortest path.
    - Using A* or one of the other search algorithms we've seen.

- Our continuous problem is now a discrete search problem.

17



## Visibility Graph Method

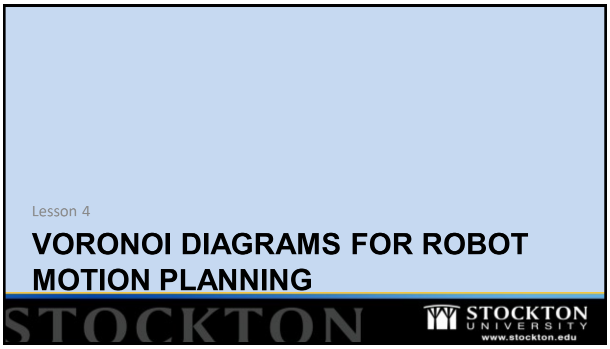
COMPLAINT

- Visibility graph method finds the shortest path.
    - Assuming you use an optimal algorithm like A* with an admissible heuristic
- But it does so by skirting along and close to obstacles.
- Any error in control, or model of obstacle locations, and Bang! Screech!!

Who cares about optimality?

Perhaps we want to get a non-stupid path that steers as far from the obstacles as it can.

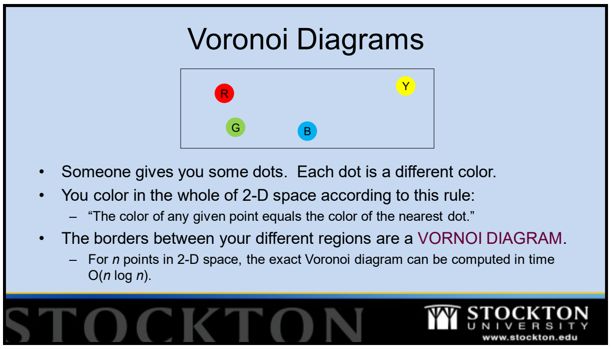
18



Lesson 4

# VORONOI DIAGRAMS FOR ROBOT MOTION PLANNING

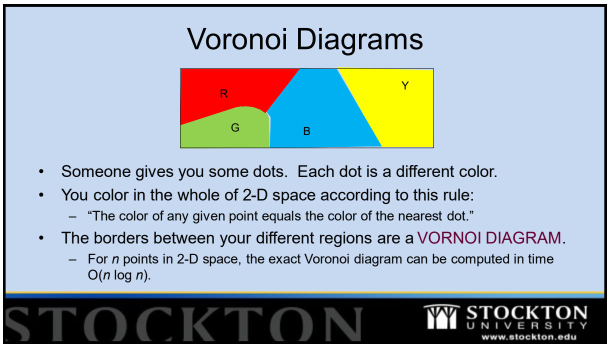
19



## Voronoi Diagrams

- Someone gives you some dots. Each dot is a different color.
- You color in the whole of 2-D space according to this rule:
  - "The color of any given point equals the color of the nearest dot."
- The borders between your different regions are a VORNOI DIAGRAM.
  - For *n* points in 2-D space, the exact Voronoi diagram can be computed in time O(*n* log *n*).

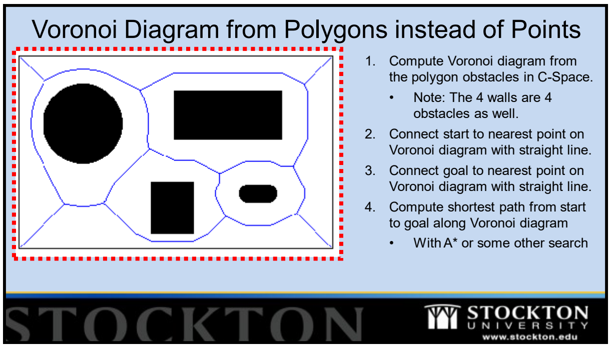
20



## Voronoi Diagrams

- Someone gives you some dots. Each dot is a different color.
- You color in the whole of 2-D space according to this rule:
  - "The color of any given point equals the color of the nearest dot."
- The borders between your different regions are a VORNOI DIAGRAM.
  - For *n* points in 2-D space, the exact Voronoi diagram can be computed in time O(*n* log *n*).

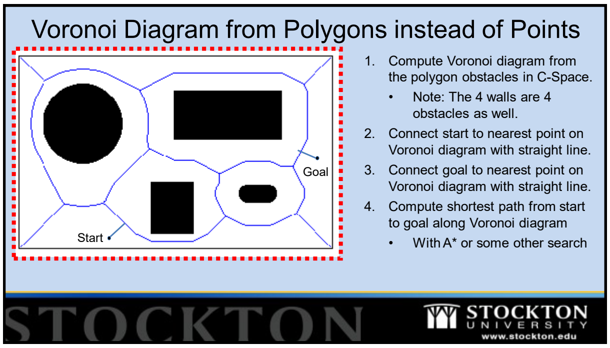
21



## Voronoi Diagram from Polygons instead of Points

1. Compute Voronoi diagram from the polygon obstacles in C-Space.
   - Note: The 4 walls are 4 obstacles as well.
2. Connect start to nearest point on Voronoi diagram with straight line.
3. Connect goal to nearest point on Voronoi diagram with straight line.
4. Compute shortest path from start to goal along Voronoi diagram
   - With A* or some other search

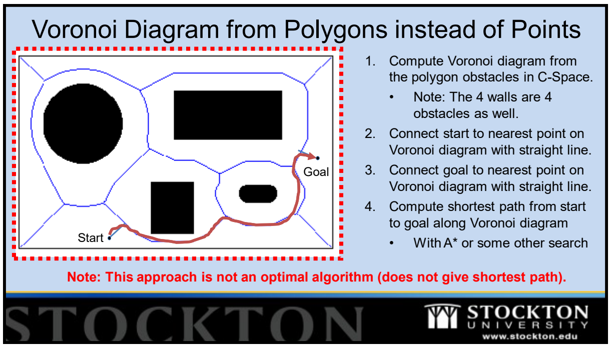
22



## Voronoi Diagram from Polygons instead of Points

1. Compute Voronoi diagram from the polygon obstacles in C-Space.
   - Note: The 4 walls are 4 obstacles as well.
2. Connect start to nearest point on Voronoi diagram with straight line.
3. Connect goal to nearest point on Voronoi diagram with straight line.
4. Compute shortest path from start to goal along Voronoi diagram
   - With A* or some other search

23

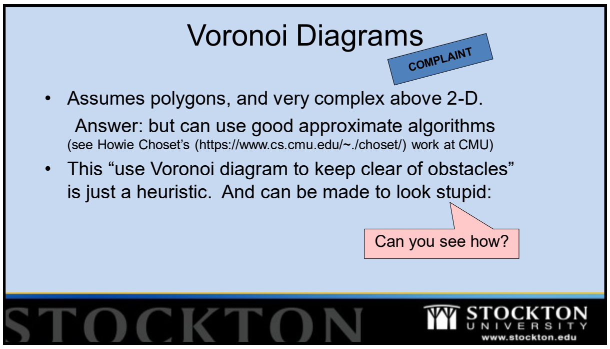


## Voronoi Diagram from Polygons instead of Points

1. Compute Voronoi diagram from the polygon obstacles in C-Space.
   - Note: The 4 walls are 4 obstacles as well.
2. Connect start to nearest point on Voronoi diagram with straight line.
3. Connect goal to nearest point on Voronoi diagram with straight line.
4. Compute shortest path from start to goal along Voronoi diagram
   - With A* or some other search

**Note: This approach is not an optimal algorithm (does not give shortest path).**

24



## Voronoi Diagrams

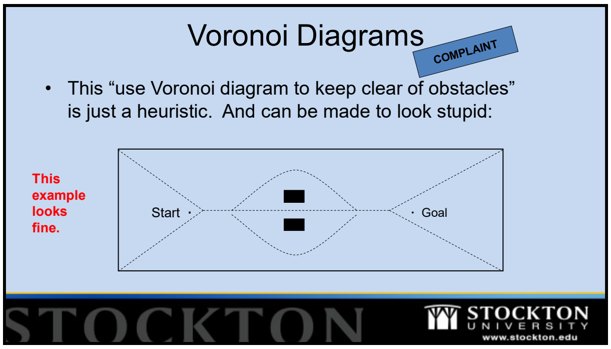
COMPLAINT

- Assumes polygons, and very complex above 2-D.

  Answer: but can use good approximate algorithms (see Howie Choset's (https://www.cs.cmu.edu/~./choset/) work at CMU)
- This "use Voronoi diagram to keep clear of obstacles" is just a heuristic. And can be made to look stupid:

Can you see how?

25



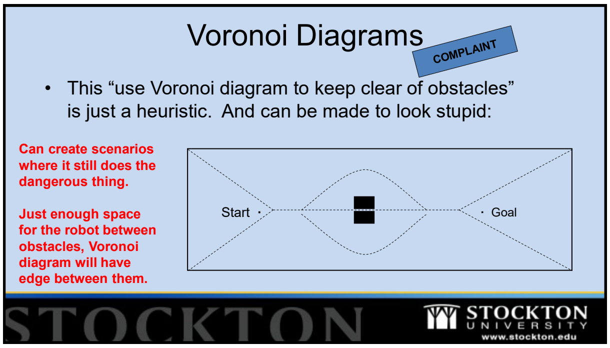
## Voronoi Diagrams

COMPLAINT

- This "use Voronoi diagram to keep clear of obstacles" is just a heuristic. And can be made to look stupid:

**This example looks fine.**

26



## Voronoi Diagrams

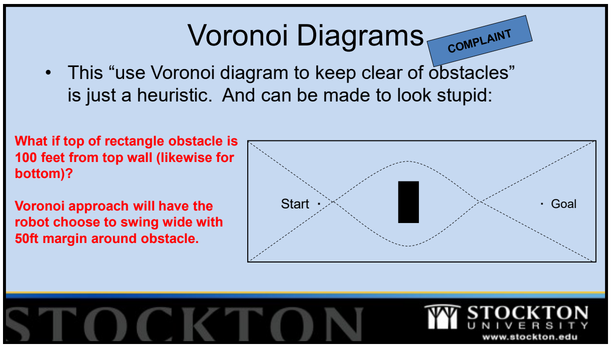
COMPLAINT

- This "use Voronoi diagram to keep clear of obstacles" is just a heuristic. And can be made to look stupid:

**Can create scenarios where it still does the dangerous thing.**

**Just enough space for the robot between obstacles, Voronoi diagram will have edge between them.**

27



## Voronoi Diagrams

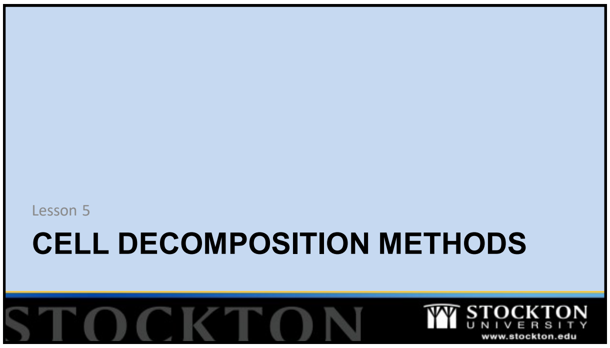
COMPLAINT

- This "use Voronoi diagram to keep clear of obstacles" is just a heuristic. And can be made to look stupid:

**What if top of rectangle obstacle is 100 feet from top wall (likewise for bottom)?**

**Voronoi approach will have the robot choose to swing wide with 50ft margin around obstacle.**

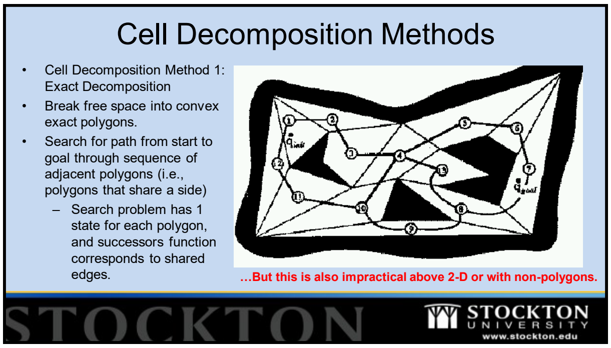
28



Lesson 5

## CELL DECOMPOSITION METHODS

29

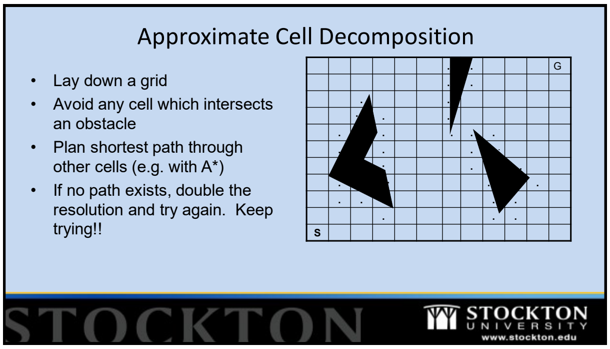


## Cell Decomposition Methods

- Cell Decomposition Method 1: Exact Decomposition
- Break free space into convex exact polygons.
- Search for path from start to goal through sequence of adjacent polygons (i.e., polygons that share a side)
  - Search problem has 1 state for each polygon, and successors function corresponds to shared edges.

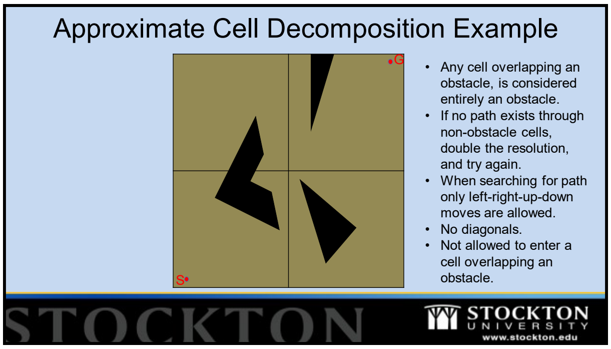
**…But this is also impractical above 2-D or with non-polygons.**

30



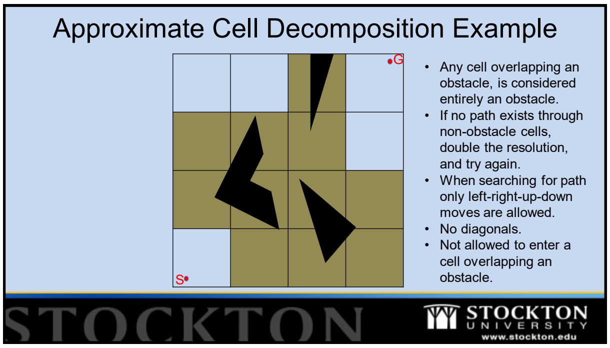
## Approximate Cell Decomposition

- Lay down a grid
- Avoid any cell which intersects an obstacle
- Plan shortest path through other cells (e.g. with A*)
- If no path exists, double the resolution and try again. Keep trying!!



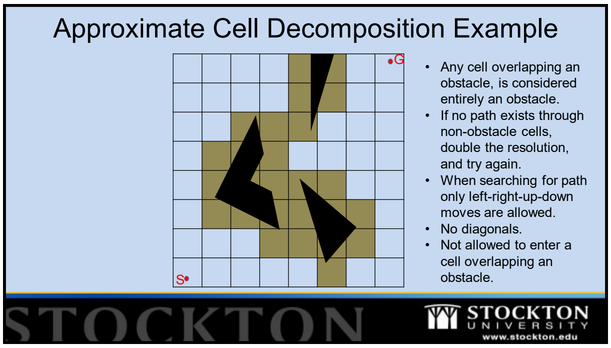
## Approximate Cell Decomposition Example

- Any cell overlapping an obstacle, is considered entirely an obstacle.
- If no path exists through non-obstacle cells, double the resolution, and try again.
- When searching for path only left-right-up-down moves are allowed.
- No diagonals.
- Not allowed to enter a cell overlapping an obstacle.



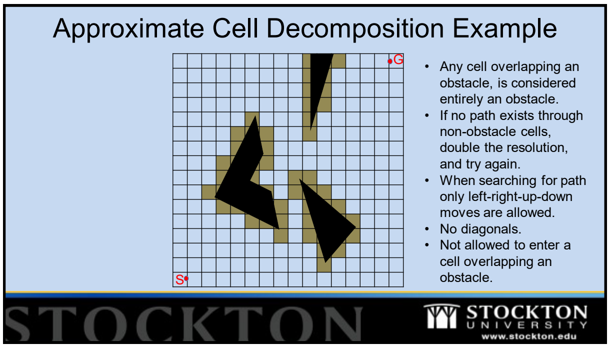
## Approximate Cell Decomposition Example

- Any cell overlapping an obstacle, is considered entirely an obstacle.
- If no path exists through non-obstacle cells, double the resolution, and try again.
- When searching for path only left-right-up-down moves are allowed.
- No diagonals.
- Not allowed to enter a cell overlapping an obstacle.



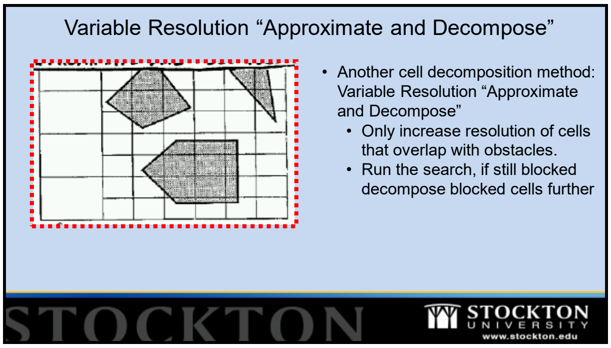
## Approximate Cell Decomposition Example

- Any cell overlapping an obstacle, is considered entirely an obstacle.
- If no path exists through non-obstacle cells, double the resolution, and try again.
- When searching for path only left-right-up-down moves are allowed.
- No diagonals.
- Not allowed to enter a cell overlapping an obstacle.



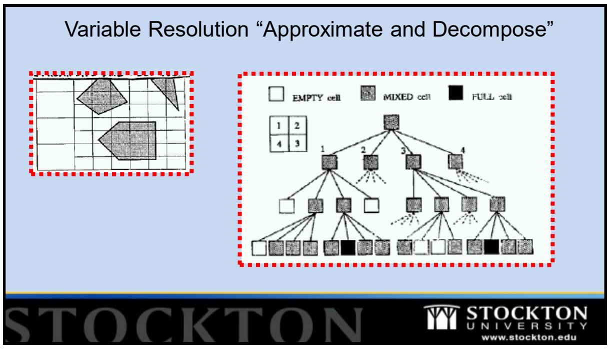
## Approximate Cell Decomposition Example

- Any cell overlapping an obstacle, is considered entirely an obstacle.
- If no path exists through non-obstacle cells, double the resolution, and try again.
- When searching for path only left-right-up-down moves are allowed.
- No diagonals.
- Not allowed to enter a cell overlapping an obstacle.



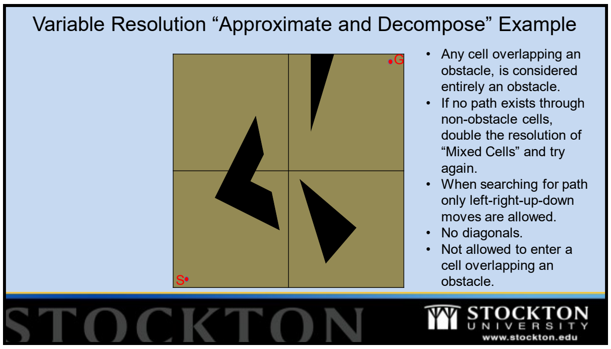
## Variable Resolution “Approximate and Decompose”

- Another cell decomposition method: Variable Resolution “Approximate and Decompose”
  - Only increase resolution of cells that overlap with obstacles.
  - Run the search, if still blocked decompose blocked cells further



## Variable Resolution “Approximate and Decompose”

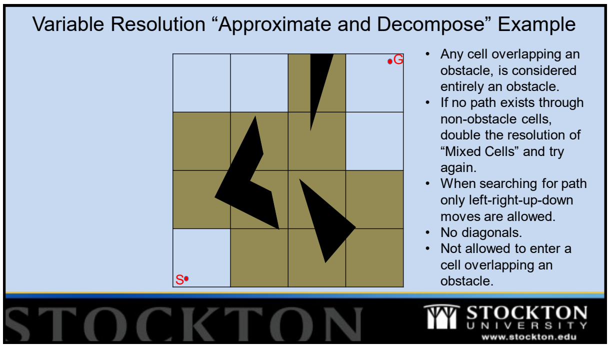
37



## Variable Resolution “Approximate and Decompose” Example

- Any cell overlapping an obstacle, is considered entirely an obstacle.
- If no path exists through non-obstacle cells, double the resolution of “Mixed Cells” and try again.
- When searching for path only left-right-up-down moves are allowed.
- No diagonals.
- Not allowed to enter a cell overlapping an obstacle.

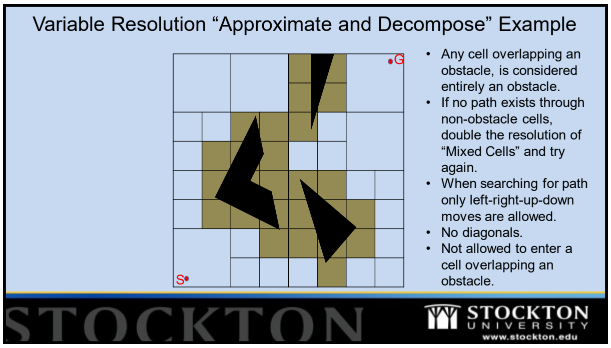
38



## Variable Resolution “Approximate and Decompose” Example

- Any cell overlapping an obstacle, is considered entirely an obstacle.
- If no path exists through non-obstacle cells, double the resolution of “Mixed Cells” and try again.
- When searching for path only left-right-up-down moves are allowed.
- No diagonals.
- Not allowed to enter a cell overlapping an obstacle.

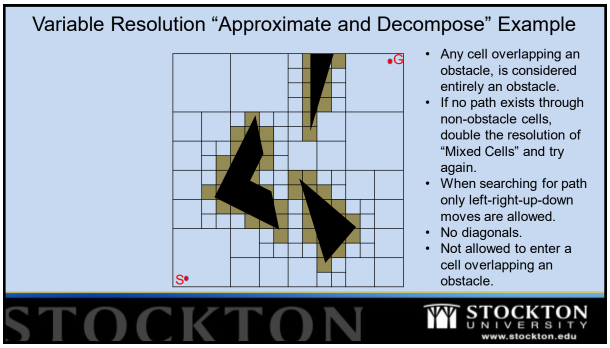
39



## Variable Resolution “Approximate and Decompose” Example

- Any cell overlapping an obstacle, is considered entirely an obstacle.
- If no path exists through non-obstacle cells, double the resolution of “Mixed Cells” and try again.
- When searching for path only left-right-up-down moves are allowed.
- No diagonals.
- Not allowed to enter a cell overlapping an obstacle.

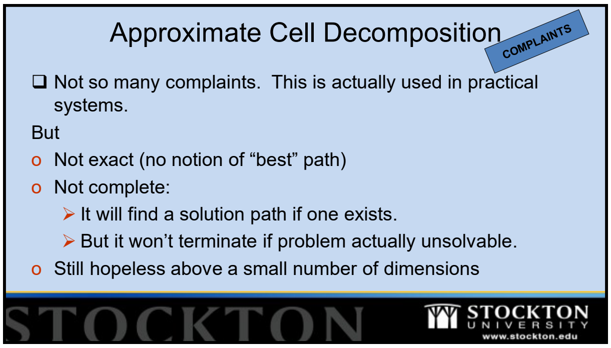
40



## Variable Resolution “Approximate and Decompose” Example

- Any cell overlapping an obstacle, is considered entirely an obstacle.
- If no path exists through non-obstacle cells, double the resolution of “Mixed Cells” and try again.
- When searching for path only left-right-up-down moves are allowed.
- No diagonals.
- Not allowed to enter a cell overlapping an obstacle.

41



## Approximate Cell Decomposition

COMPLAINTS

❑ Not so many complaints. This is actually used in practical systems.

But

- Not exact (no notion of “best” path)
- Not complete:
  - It will find a solution path if one exists.
  - But it won’t terminate if problem actually unsolvable.
- Still hopeless above a small number of dimensions

42

Lesson 6

## POTENTIAL FIELDS

43

## Potential Fields for Robot Motion Planning

- Using potential fields is a very different approach.
- No search algorithms.
- Create a model that has repulsive forces
  - Use sensors to detect distances to nearby obstacles
  - Model has repulsive forces that "push" you away from obstacles
- Model also has an attractive force in direction of goal location
- Think of it like a landscape
  - The closer you are to the goal the lower the altitude
  - The closer you are to an obstacle the higher the altitude
- Take a small step in direction of steepest descent on this hypothetical landscape, repeat….

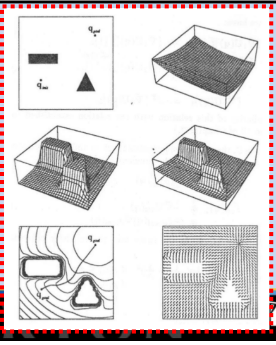
44

## Potential Fields Method

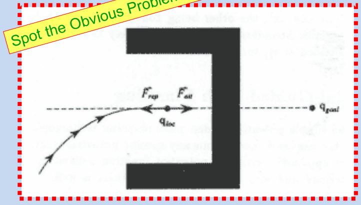
- Define a potential field function, $u(q)$, such that:
  - $u\text{: Configurations } \rightarrow \mathbb{R}$
  - Where $u(q) \rightarrow \text{huge}$, as q approaches an obstacle
  - and $u(q) \rightarrow \text{small}$, as q approaches the goal
- There is a variety of forms used for this function.
  - Google for some examples
  - Not expecting you to know the specific functions used
- Visual example next….

45

## Potential Field Example



46



Spot the Obvious Problem!

Solution I:

Use special local-minimum-free potential fields (Laplace equations can do this) – But very expensive to compute

Solution II:

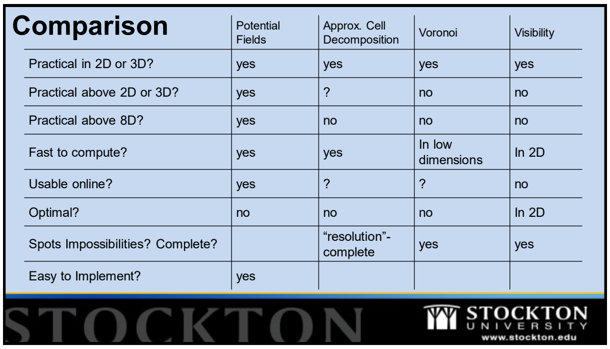
When at a local minimum start doing some searching – e.g., a brief random walk to try to escape local minima, repeat with longer random walk if get stuck again, etc.…

47

Lesson 7

## COMPARISON OF THE APPROACHES

48

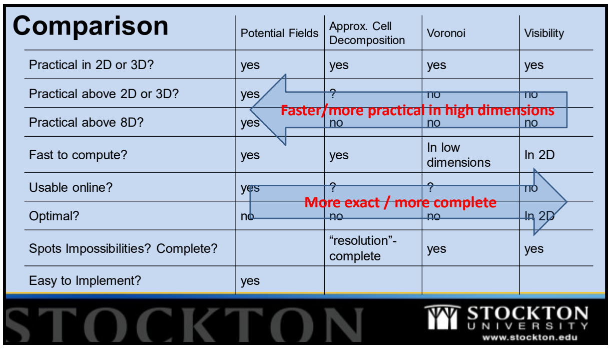


## Comparison

| | Potential Fields | Approx. Cell Decomposition | Voronoi | Visibility |
|---|---|---|---|---|
| Practical in 2D or 3D? | yes | yes | yes | yes |
| Practical above 2D or 3D? | yes | ? | no | no |
| Practical above 8D? | yes | no | no | no |
| Fast to compute? | yes | yes | In low dimensions | In 2D |
| Usable online? | yes | ? | ? | no |
| Optimal? | no | no | no | In 2D |
| Spots Impossibilities? Complete? | | “resolution”-complete | yes | yes |
| Easy to Implement? | yes | | | |

STOCKTON UNIVERSITY www.stockton.edu

49



## Comparison

| | Potential Fields | Approx. Cell Decomposition | Voronoi | Visibility |
|---|---|---|---|---|
| Practical in 2D or 3D? | yes | yes | yes | yes |
| Practical above 2D or 3D? | yes | ? | no | no |
| Practical above 8D? | yes | no | no | no |
| Fast to compute? | yes | yes | In low dimensions | In 2D |
| Usable online? | yes | ? | ? | no |
| Optimal? | no | no | no | In 2D |
| Spots Impossibilities? Complete? | | “resolution”-complete | yes | yes |
| Easy to Implement? | yes | | | |

Faster/more practical in high dimensions

More exact / more complete

STOCKTON UNIVERSITY www.stockton.edu

50