MOBILE ROBOTS 4/5 - Navigation

Prof. Francesco Mondada

EPFL

2025-2026

Navigation

What is navigation? Finding a collision-free path from one pose to another

Navigation algorithm properties:
• **Optimality:** does the planner find trajectories that are optimal in some sense (length, execution time, energy consumption)?
• **Completeness:** does the planner always find a solution when one exists?
• **Offline / online:** Can the solution be computed in real time or is too heavy computationally?

Questions: Do we have a model of the environment (map) or only the direction to the goal? If a map exists, are all obstacles included? Do we need to take into account geometry, kinematics constraints, and/or the dynamics of the robot?

Note that most existing techniques make the **assumption of a mass-less, holonomic, point-like robot** => may require low-level motion control and a priori expansion of obstacles to be implemented robustly

Navigation : Local versus Global

Local : Obstacle Avoidance

• Tactical: modulating the trajectory to avoid unforeseen, local obstacles
• No map or only local and sensor-based
• Rather reactive: no complex sensor processing => fast

Necessary in mobile robotics because environments are not fully predictable (uncertainty in sensors and maps, as well as presence of dynamic obstacles)

Global :Path Planning (or motion planning)

• Strategic: planning the global trajectory
• Given a map (metric, grid-based or topological) and a goal location
• Rather cognitive: planning of a series of actions => time consuming

Compared to industrial robotics :

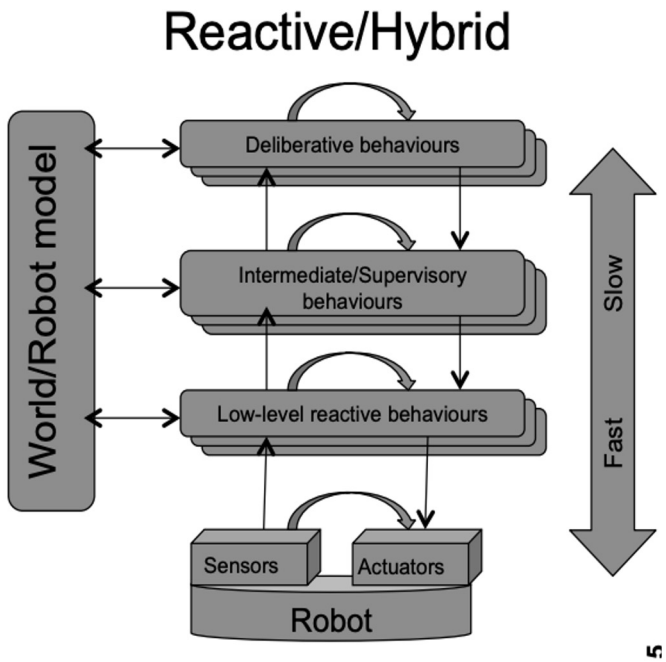
- less complicated (less DOF) but more frequent (discrepancies map vs. real environment)

Architectures – Robotic Architecture Styles

from lesson 1

path planning →

obstacle avoidance →



1. Deliberative planning to decompose the task into sub tasks
2. execute behaviours reactively, which is to say, in a succession of sense act couplings

Sensor data gets routed to each behaviour that needs that sensor, but is also available to the planner for construction of a task-oriented global world model.

Obstacle Avoidance (local navigation)

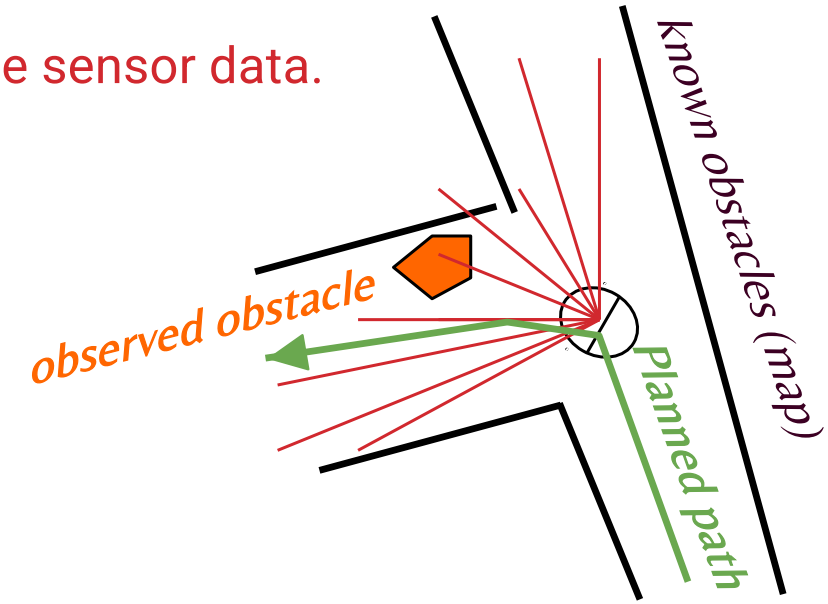
The goal of obstacle avoidance algorithms (as opposed to path planning) is to avoid collisions with unpredictable objects or due to uncertainty in the localization process or the map.

It is either based on a *local map* or directly uses the sensor data.

It is often implemented as an independent task running at high frequency.

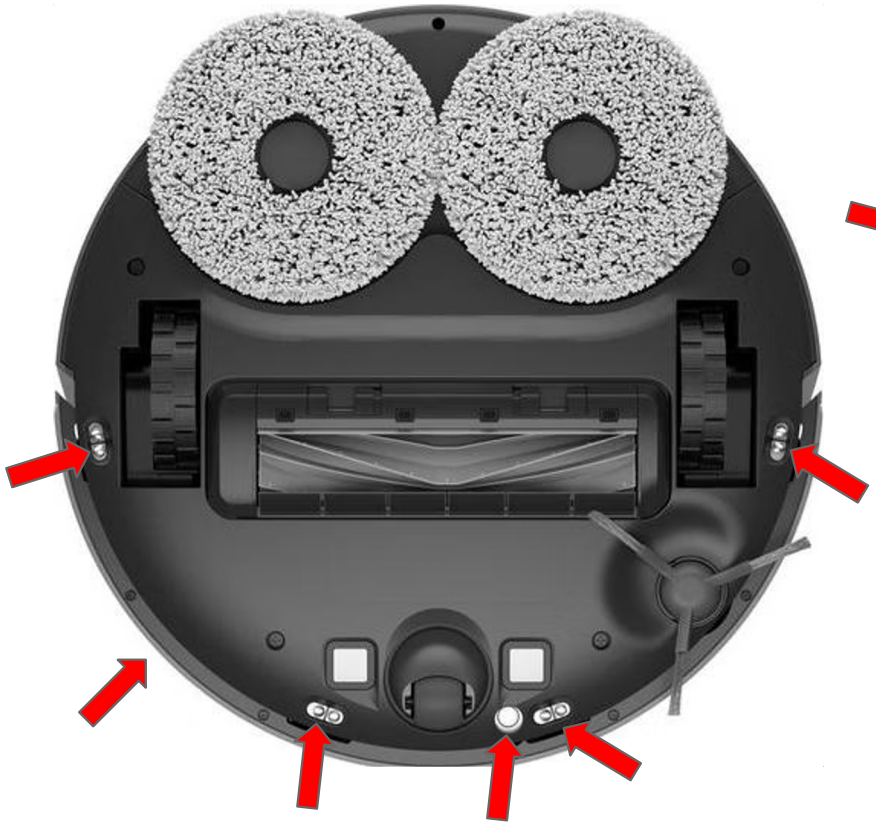
Efficient obstacle avoidance methods should take into account:

- the sensor readings
- the kinematics / dynamics of the robot
- the overall goal direction



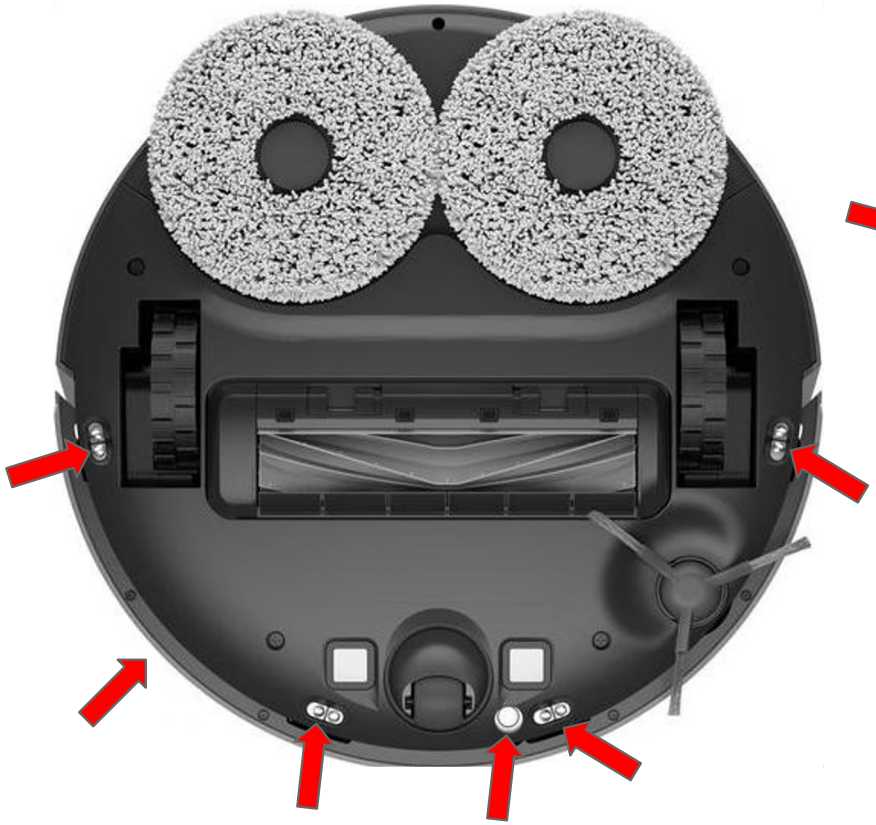
What type of sensors are used?

An example: Dreame L10S Pro. Which one for local navigation? (here all the sensors)



What type of sensors are used?

An example: Dreame L10S Pro. Which one for local navigation? Nearly all



Obstacle Avoidance – Tactile Sensors

Mainly used for two purposes, based on observation of nature:

- **Manipulation**

  measurement of point and force of contact with objects that are manipulated, definition of stable positions or detection of dynamic situations,

- **Exploration**

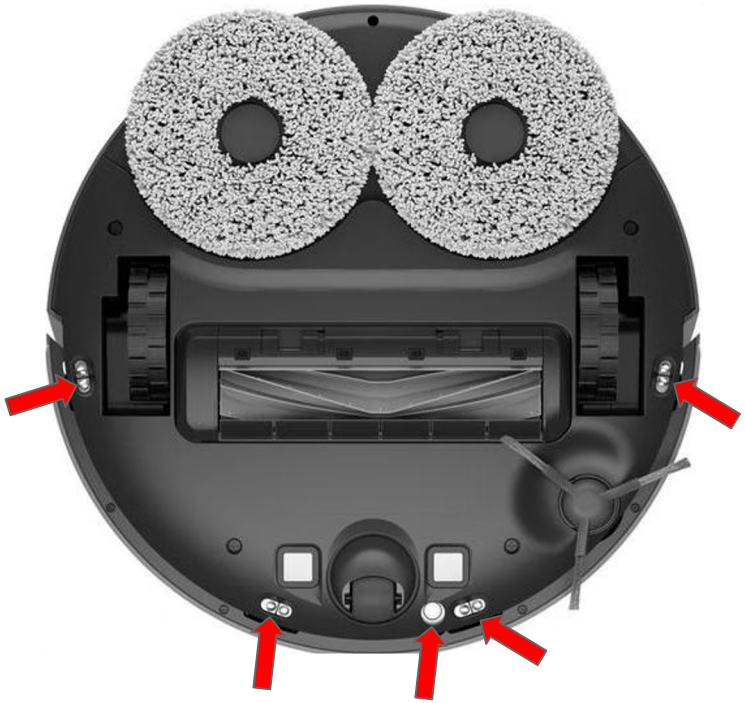
  detection of close objects, as well as some physical properties like surface texture, hardness, temperature, friction coefficients, etc.

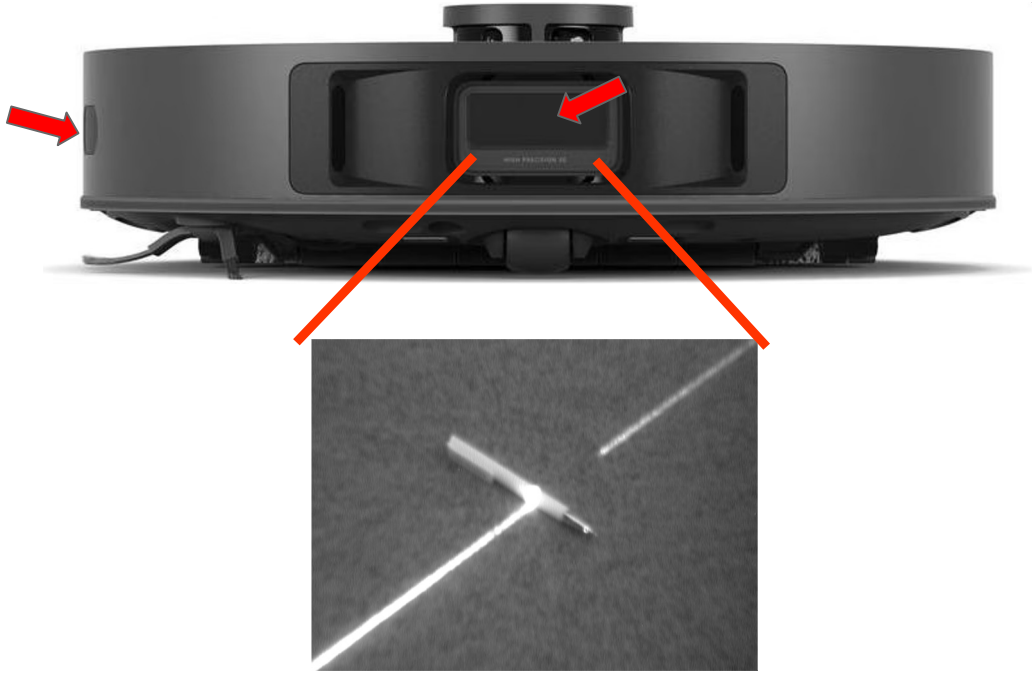
Tactile Sensors Based on Normal Pressure:

- Contact closure
  - Only binary
  - Used for security purposes
- Piezoresistive
  - Measures forces
  - Signal drifts and has hysteresis
- Piezoelectric
  - Measures forces
  - High bandwidth
  - Critical electrical junctions
- Capacitive
  - Measures forces
  - Complex circuitry

Obstacle Avoidance – Range Sensors

- Range information:
  - key element for obstacle avoidance, localization and environment modeling





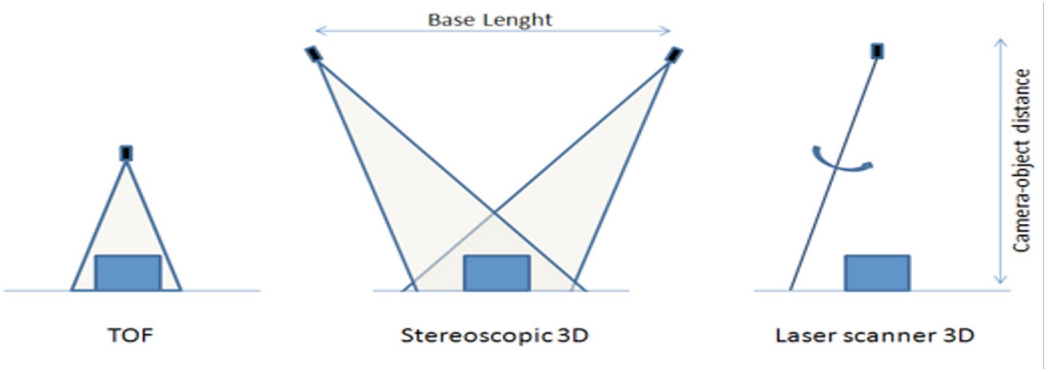
Obstacle Avoidance – Time Of Flight Sensors

- Range information:
  - key element for obstacle avoidance, localization and environment modeling
- Ultrasonic sensors and laser range finders make use of propagation speed of sound or electromagnetic waves respectively. The traveled distance of a sound or electromagnetic wave is given by

$$l = c \cdot t$$

Where

- l = distance traveled (usually round-trip)
- c = speed of wave propagation
- t = time of flight



Obstacle Avoidance – Time Of Flight Sensors

Important to keep in mind:

- Propagation speed of sound: 0.3m/**ms**
- Propagation speed of of electromagnetic signals: 0.3m/**ns**
- 3 meters correspond to 10ms for an ultrasonic system versus only 10ns for a laser range sensor

  => time of flight with electromagnetic signals involves very fast electronics

  => laser range sensors are more expensive and delicate to design

The quality of TOF range sensors mainly depends on:

- Uncertainties about the exact time of arrival of the reflected signal
  - Inaccuracies in the time of fight measure (laser range sensors)
- Opening angle of transmitted beam (especially ultrasonic range sensors)
- Interaction with the target (surface, diffuse/specular reflections)
- Variation of propagation speed (sound)
- Speed of mobile robot and target (if not at stand still)

TOF Sensors – Ultrasonic Sensors

Transmit a packet of (ultrasonic) pressure waves

Distance *d* of the echoing object can be calculated based on the propagation speed of sound *c* and the time of flight *t*.

$$d = \frac{c \cdot t}{2}$$

The speed of sound *c* in air is given by

$$c = \sqrt{\gamma \cdot R \cdot T}$$

where

γ : ratio of specific heats or adiabatic index (1.402 for air)

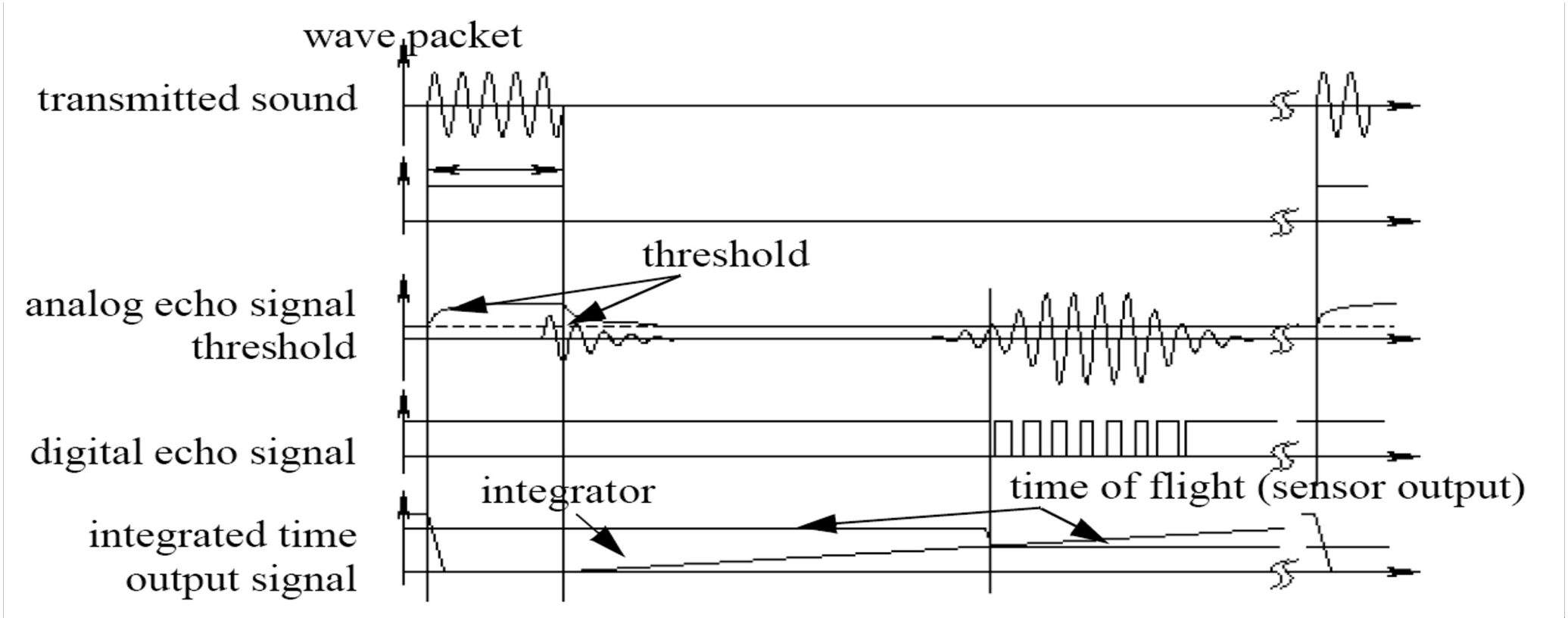
R: gas constant (287.05 $J \cdot kg^{-1} \cdot K^{-1}$ for air)

T: temperature in Kelvin

In air at standard pressure and 20° Celsius the speed of sound is around c = 343 m/s.

TOF Sensors – Ultrasonic Sensors

Possible implementation (very basic scheme)



TOF Sensors – Ultrasonic Sensors

Typical frequency: 40 - 180kHz

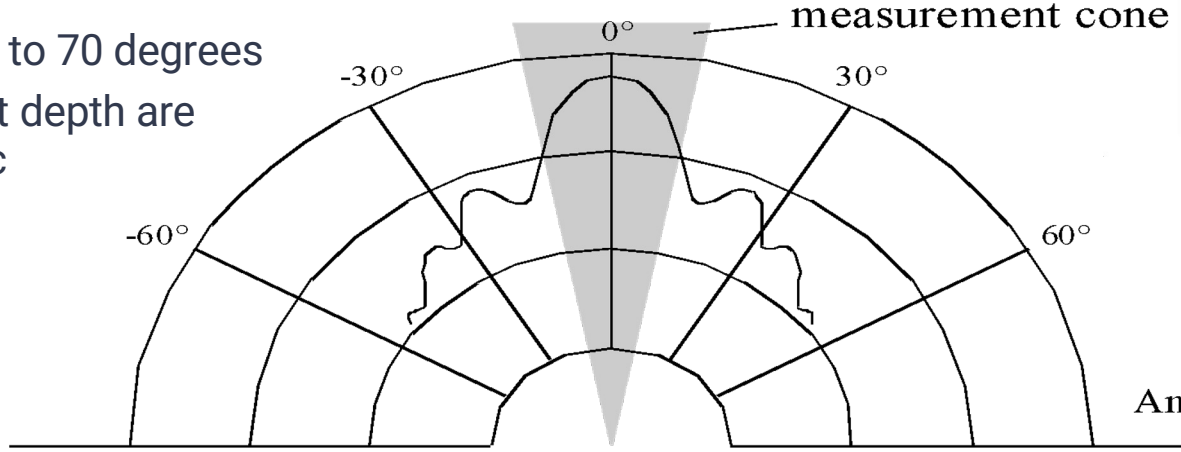
- typical range: 12 cm to 5 m; resolution: ~2 cm

Generation of sound wave: piezo transducer

- transmitter and receiver can be separated or not

Sound beam propagates in a cone (*approx.*)

- opening angles: 20 to 70 degrees
- regions of constant depth are segments of an arc (sphere for 3D)

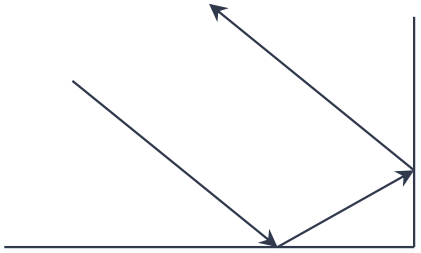


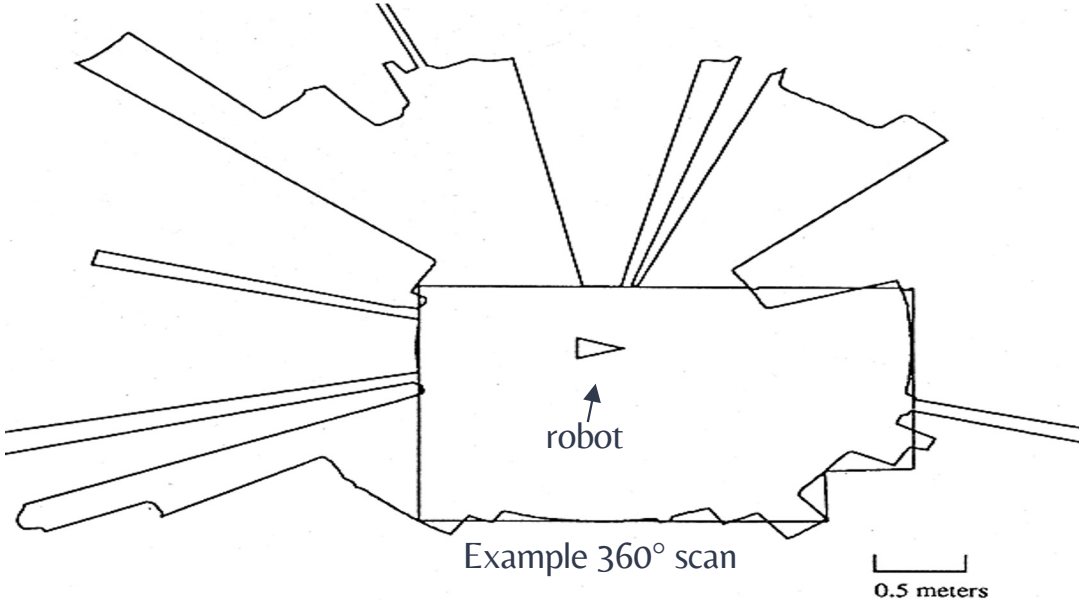
Typical intensity distribution of a ultrasonic sensor

TOF Sensors – Ultrasonic Sensors

Other limitations of ultrasonic sensors

- soft surfaces that absorb most of the sound energy
- surfaces that are far from being perpendicular to the direction of the sound
  => specular reflection
- blanking time errors
- Multipath errors
  - E.g. in corners



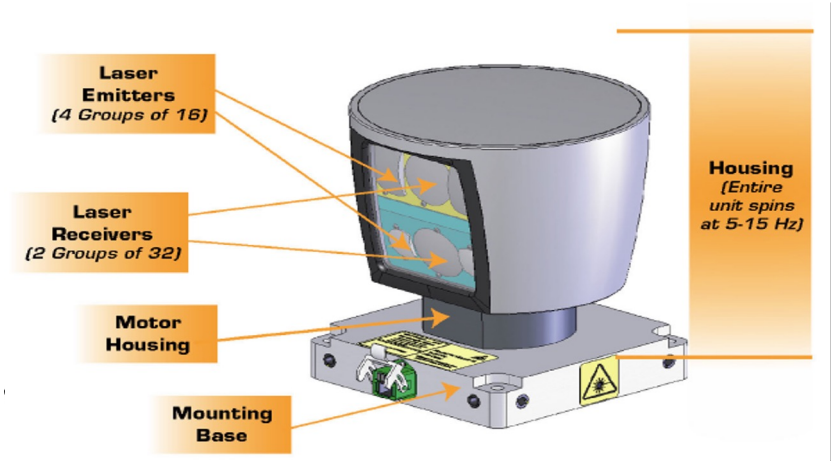


TOF Sensors – Laser RangeFinders

Time of flight measurement is generally achieved using one of the following methods:

- Pulsed laser (e.g. Sick Laser rangefinder)
  - direct measurement of time of flight
  - requires resolving picoseconds (3m = 10ns)

- Phase shift measurement (e.g. Hokuyo laser rangefinder)
  - sensor transmits 100% amplitude modulated light at a known frequency and measures the phase shift between the transmitted and reflected signals
  - technically easier than the above method because only a light intensity needs to be measured



Sick Inc., Germany
Velodyne Inc., CA

Because laser measure a point, laser are used on scanners

TOF Sensors – Laser RangeFinders : Phase Shift

Also known as *laser radar* or *LIDAR (LIght Detection And Ranging)*

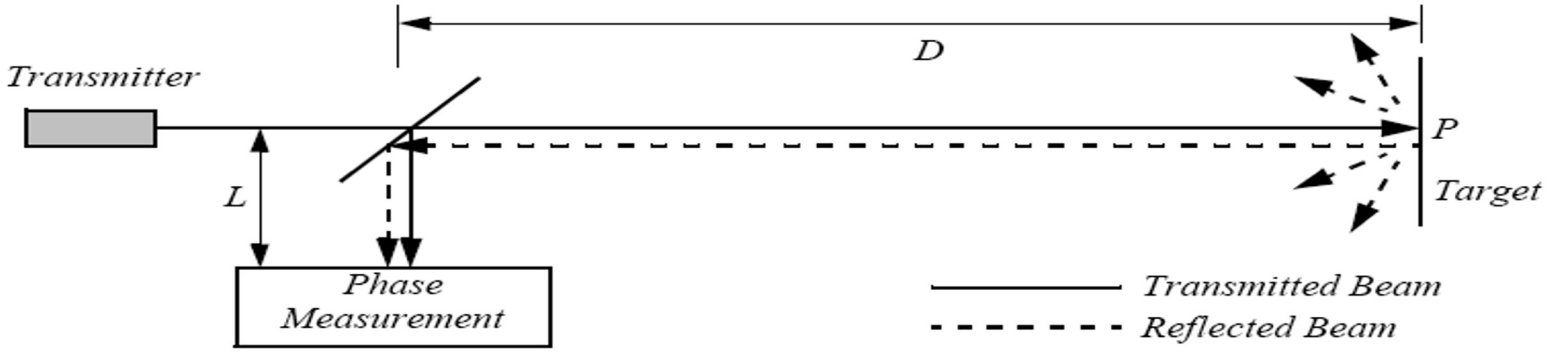
Transmitted and received beams are coaxial

Transmitter illuminates a target with a collimated beam (laser)

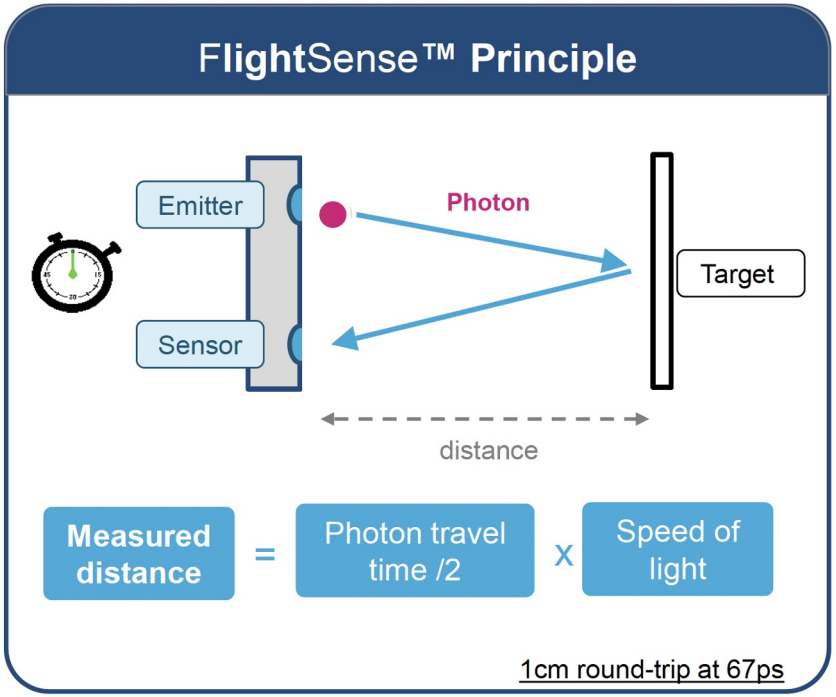
Diffuse reflection with most surfaces

Receiver detects the time needed for round-trip

An optional mechanism sweeps the light beam to cover the required scene (in 2D or 3D).

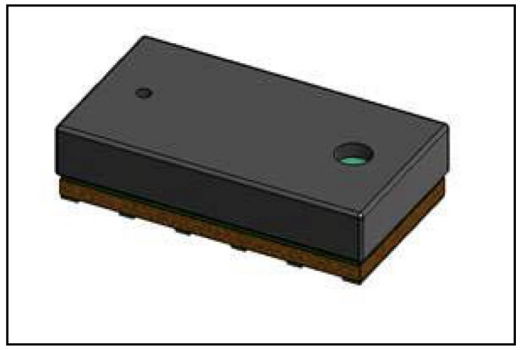


TOF Sensors – Laser RangeFinders : TOF



life.augmented

VL53L0X

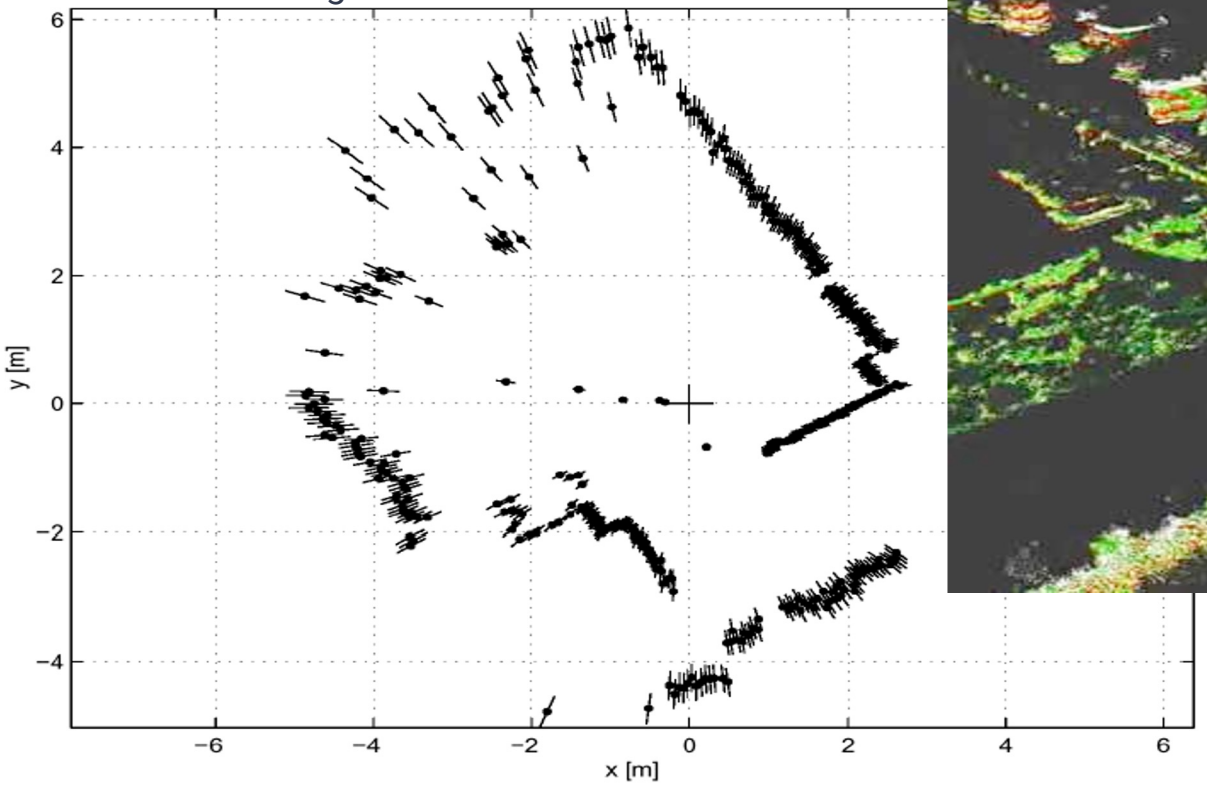


Features

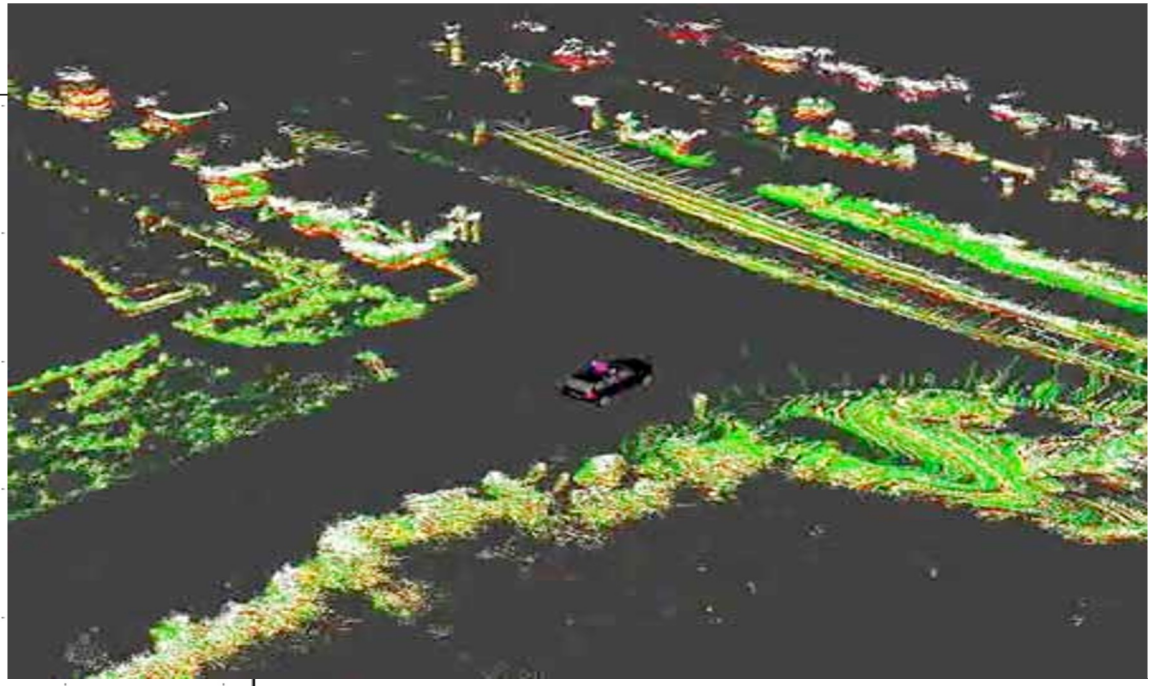
- Fully integrated miniature module
  - 940nm Laser VCSEL
  - VCSEL driver
  - Ranging sensor with advanced embedded micro controller
  - 4.4 x 2.4 x 1.0mm

TOF Sensors – Laser RangeFinders Outputs

Sick laser range finder



Velodyne LIDAR

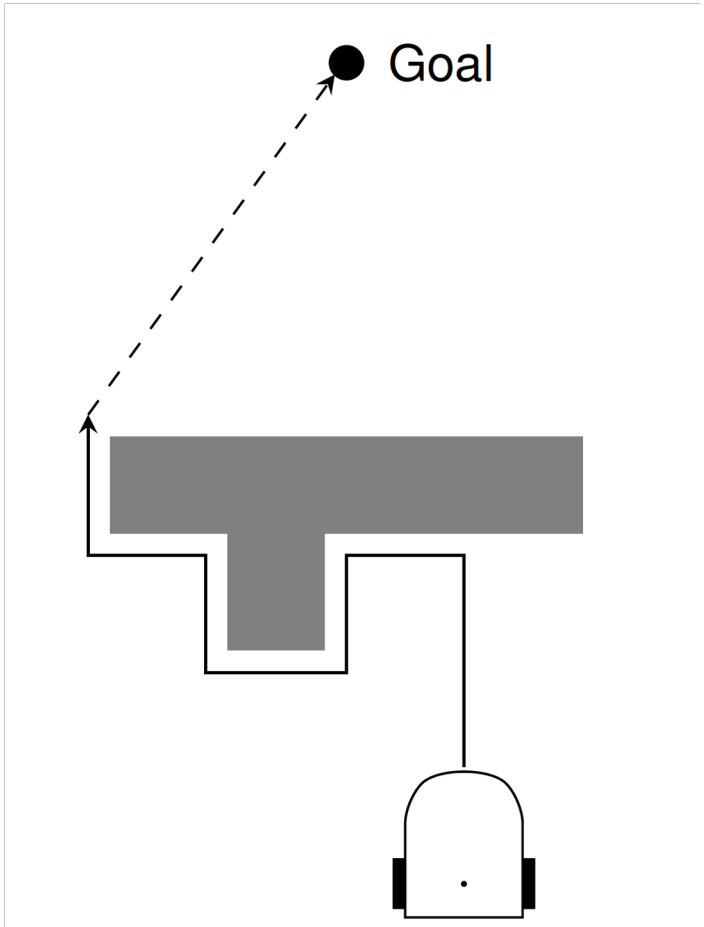


Obstacle Avoidance Strategies – Example 1

Assuming imprecise localization (distance and bearing to **visible** goal) and only proximity sensing.

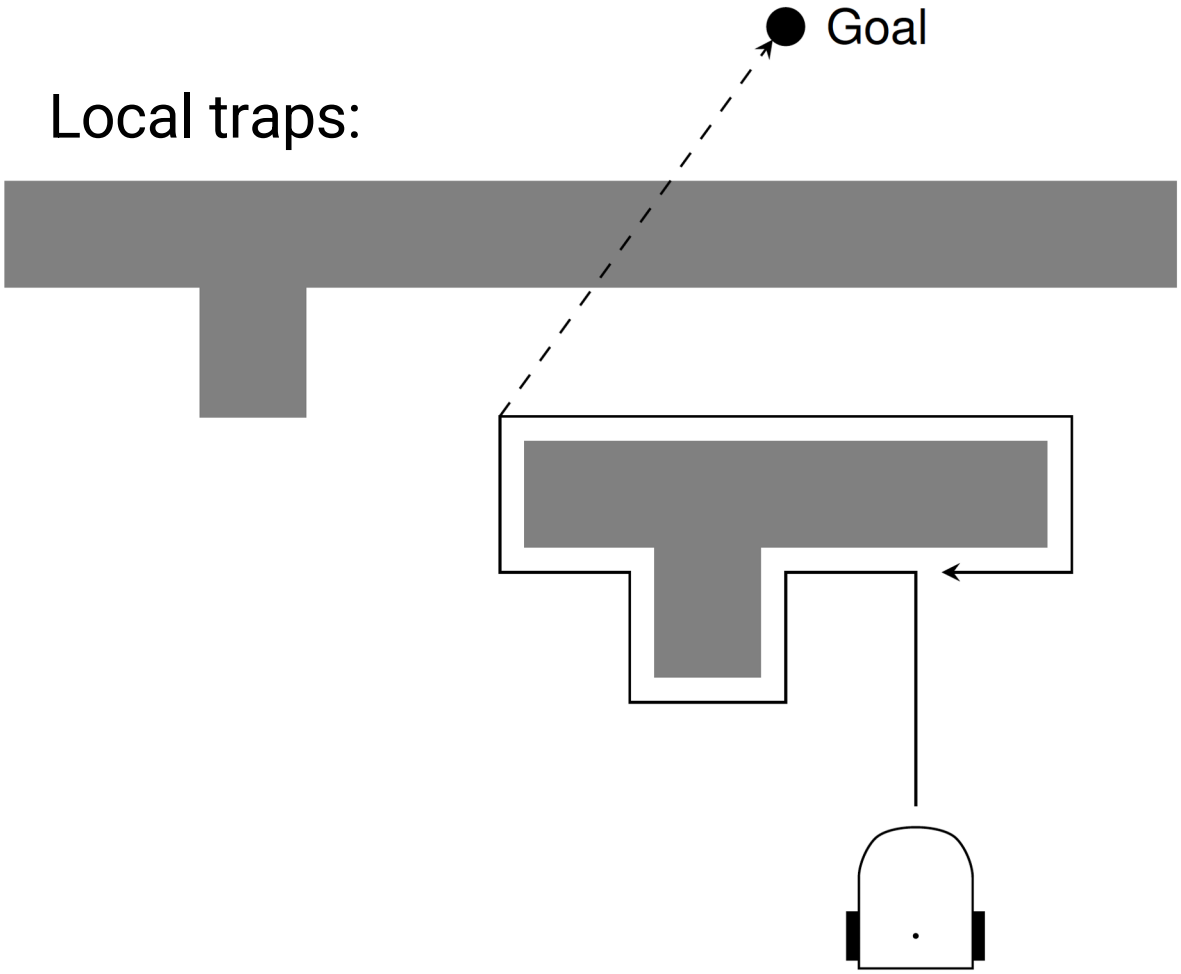
Strategy:

- Following the obstacle to avoid collision with it.
- Direction to the goal as soon as is visible again.



Obstacle Avoidance – Example 1 Problems

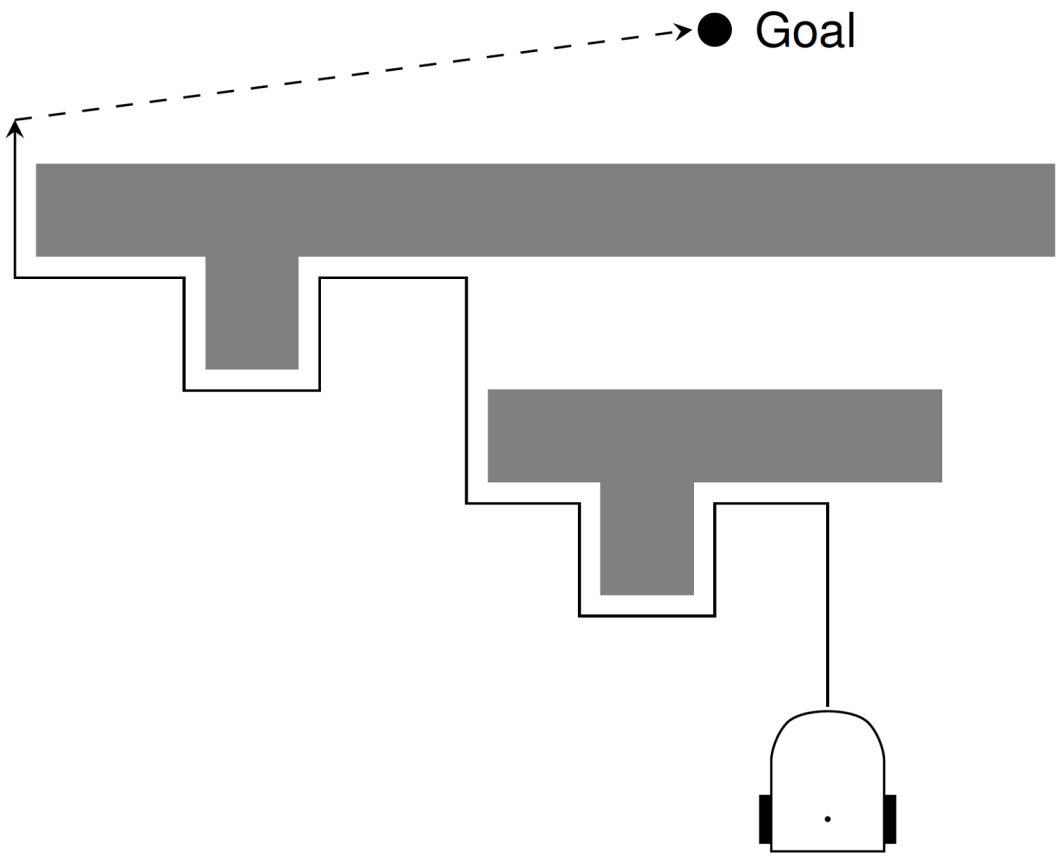
Local traps:



Obstacle Avoidance Strategies – Example 2

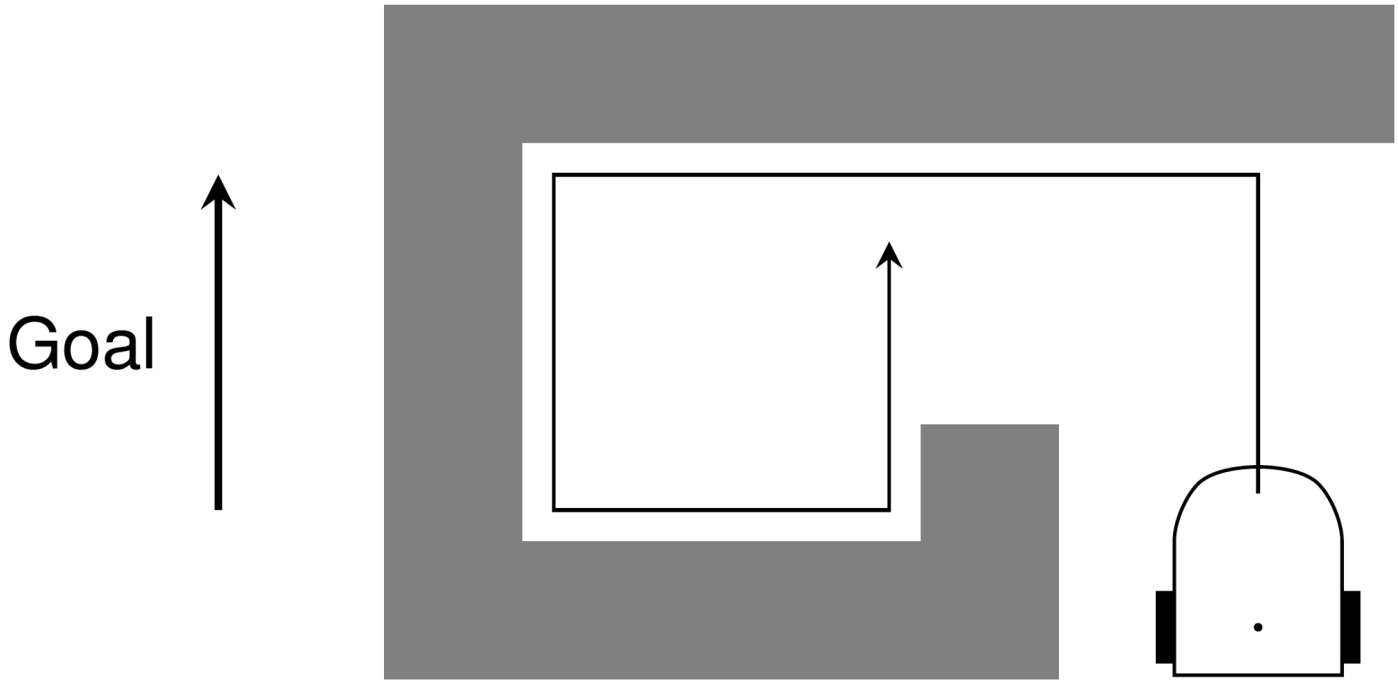
Assuming imprecise localization (distance and **continuous bearing** to goal) and only proximity sensing.

- Following the obstacle to avoid collision with it.
- Direction to the goal as soon as is **free** again. Direction can be updated by odometry. Direction is good when is a multiple of 360 degrees



Obstacle Avoidance - Example 2 Problems

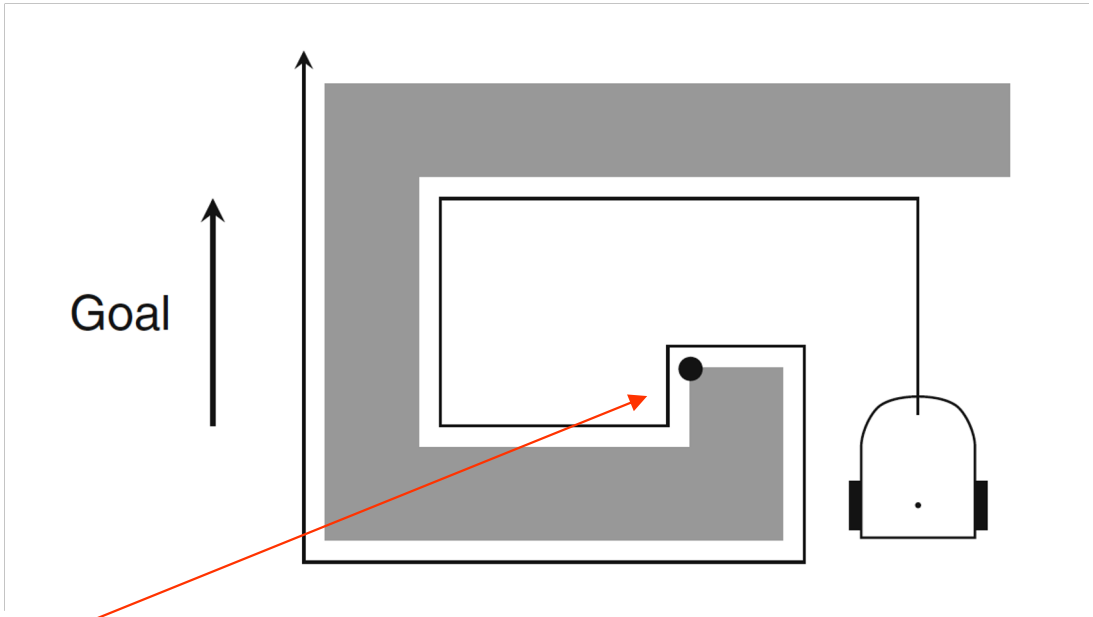
Local trap:



Obstacle Avoidance – The pledge algorithm

Assuming imprecise localization (distance and **continuous bearing** to goal) and only proximity sensing.

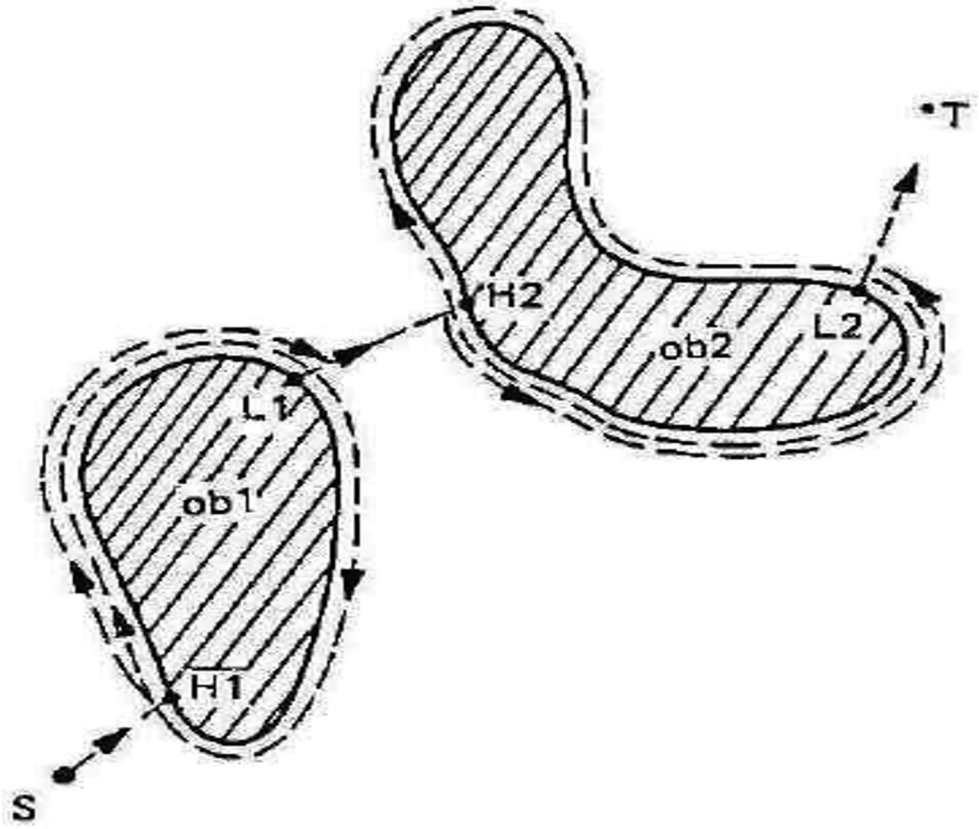
- Following the obstacle to avoid collision with it.
- Go toward the goal as soon as no obstacle is in this direction. The angle between forward direction and goal can be updated by odometry. Go toward goal **when angle is again = 0 (not +360 or -360)**



Obstacle Avoidance Strategies – Example 3

Assuming imprecise localization (distance and bearing to goal) and only proximity sensing.

- Following the obstacle to avoid collision with it.
- Each encountered obstacle is first fully circled before it is left at the point closest to the goal.



Obstacle Avoidance Strategies – Example 4

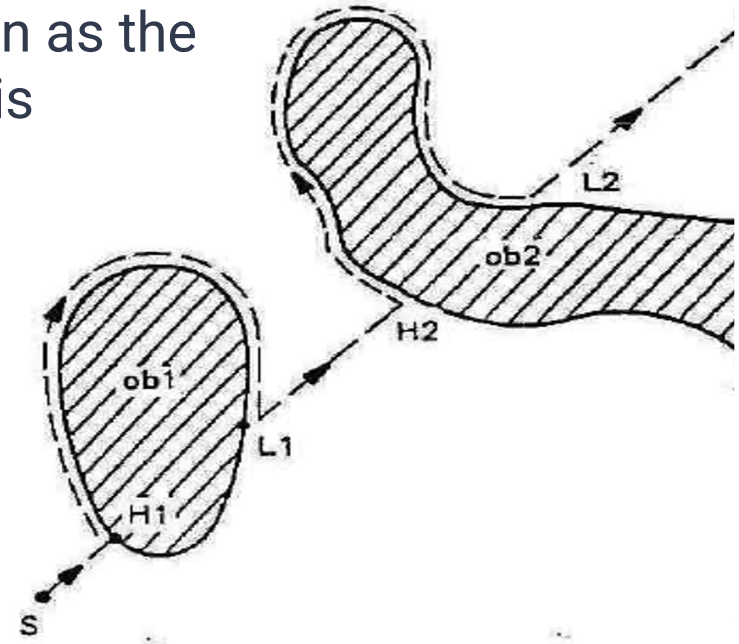
Following the obstacle always on the left (or right) side.

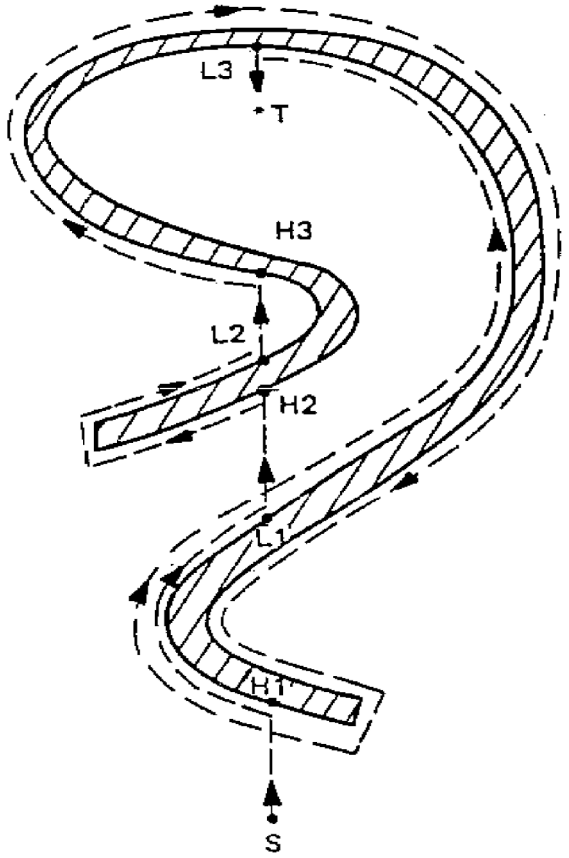
Leaving the obstacle as soon as the line between start and goal is crossed.

+ More efficient than previous.

- Requires knowing the line!

- Inefficient situations still exist.





Obstacle Avoid. – Local Mapless Potential Field

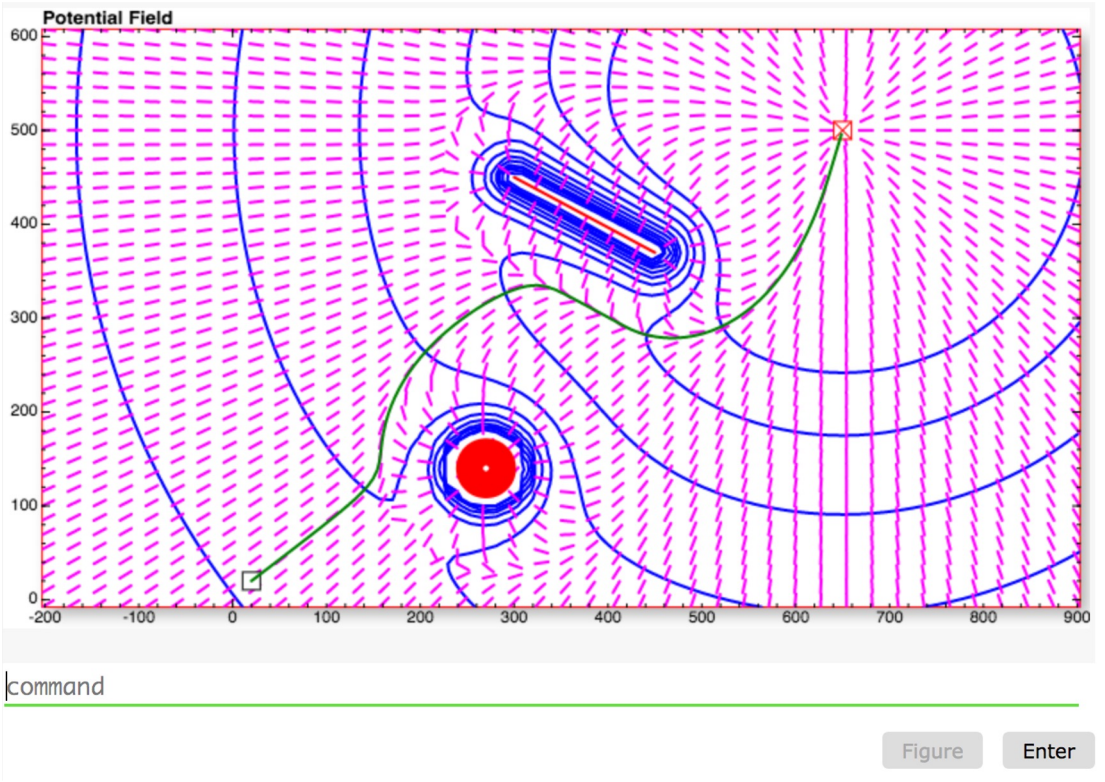
Treats range readings as a repulsive force vector, which will act on the robot (transformation from force to wheel speeds depends on kinematics).

Example of processing:

1. If the magnitude of the sum of the repulsive forces exceeds a certain threshold, the robot stops, turns into the direction of the resultant force vector, and moves on.
2. As soon as an object is detected, the repulsive force is influencing the direction of the robot

Problem of combination between attraction to a target and repulsion from the obstacles (see global navigation).

Demonstration on Potential Field



Local Occupancy Grid

Each cell represents the confidence of the algorithm in the existence of an obstacle at that location, for example:

a) For each range reading, the cell that lies on the acoustic axis and corresponds to the measured distance *d* is incremented, increasing the certainty value of the cell.

b) A histogramic pseudo-probability distribution is obtained by continuous and rapid sampling of the sensors while the robot is moving.

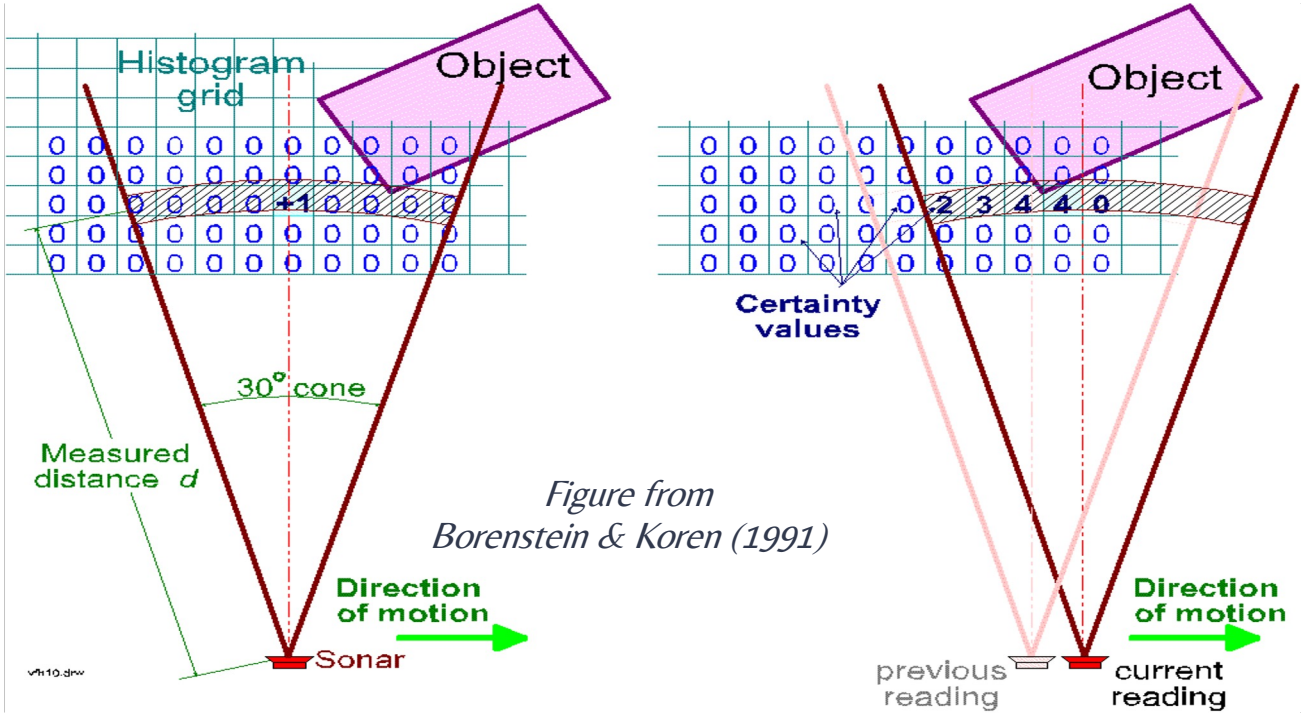


Figure from Borenstein & Koren (1991)

Curvature Velocity Method (CVM)

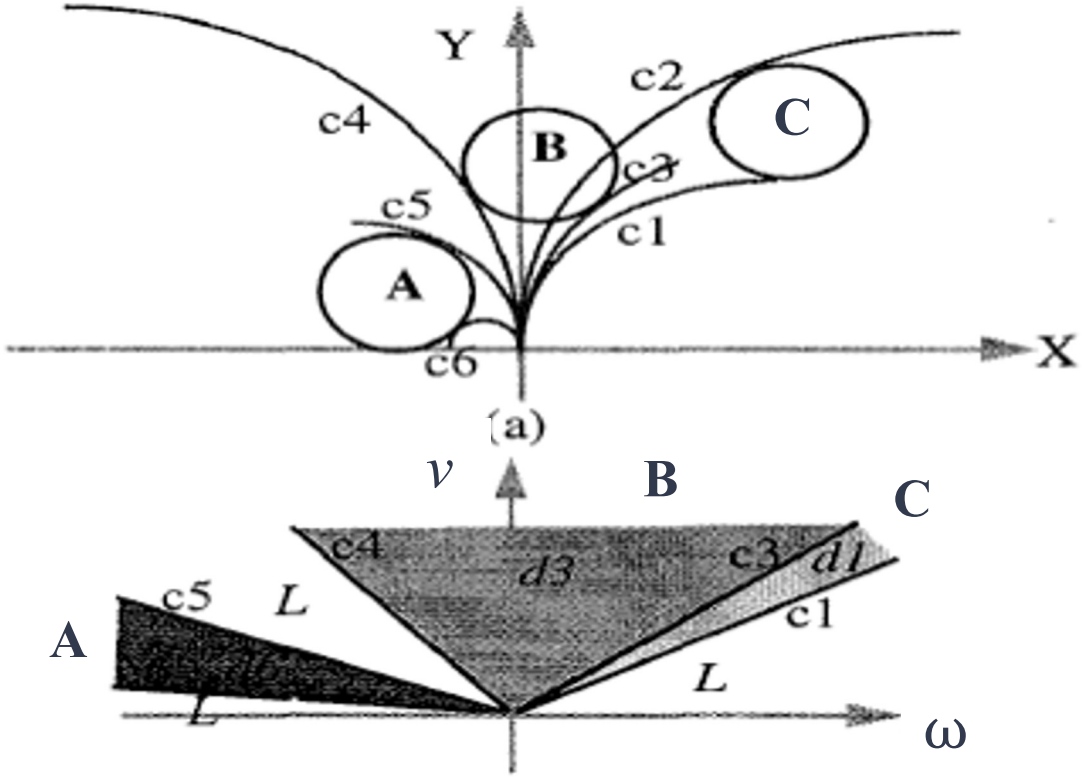
Adds *physical constraints* from the robot and the environment on the *velocity space* (*v*,ω) of the robot.

Assuming that robot is traveling on arcs (c=ω/*v*), which is a good approximation for many wheeled robots.

Obstacles are transformed from a local occupancy grid into the velocity space.

The robot chooses velocity commands that satisfy all the constraints and maximize an objective function that trades off speed, safety and goal directedness.

+ Solution corresponds directly to the commands sent to the robot.



What should I remember

- Main definition of navigation, main properties of an algorithm
- Global vs local: definitions
- Local navigation
  - main characteristics
  - types of sensors used and their main features
  - strategies
    - simplistic approaches and resulting traps
    - the pledge algorithm
    - approaches having global trajectory
    - potential field with its limitations

Path Planning : What do you need?

Global Map of the environment to know which positions are accessible or not

Start and End Position to search for the optimal path within the map

Path Planning Algorithm to find the optimal path within the map

Path Following Module to follow the path from the start to the end goal. This requires knowing the position of the robot in order to adjust the motion of the robot. The controller determines the “quality” of the line following

Local navigation module for reactive avoidance

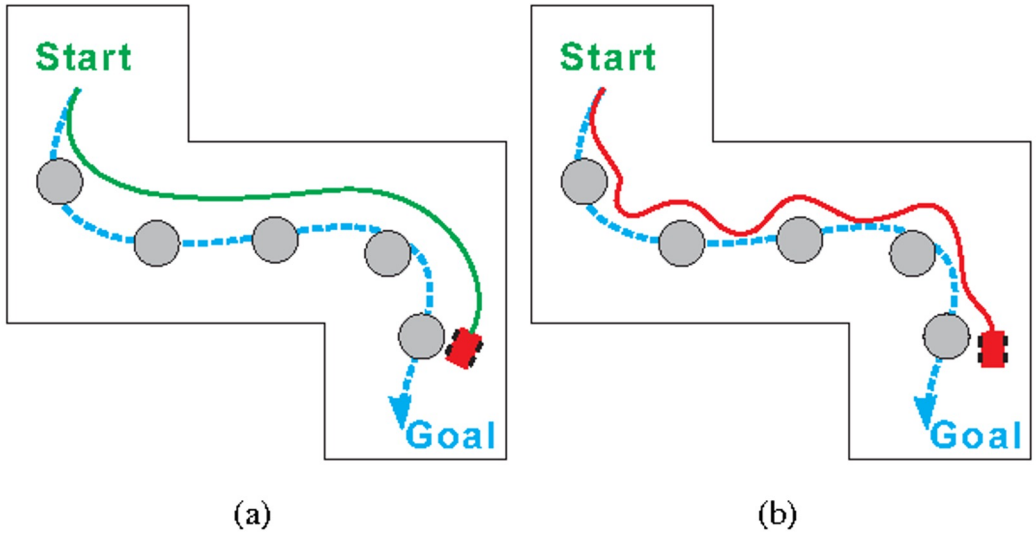


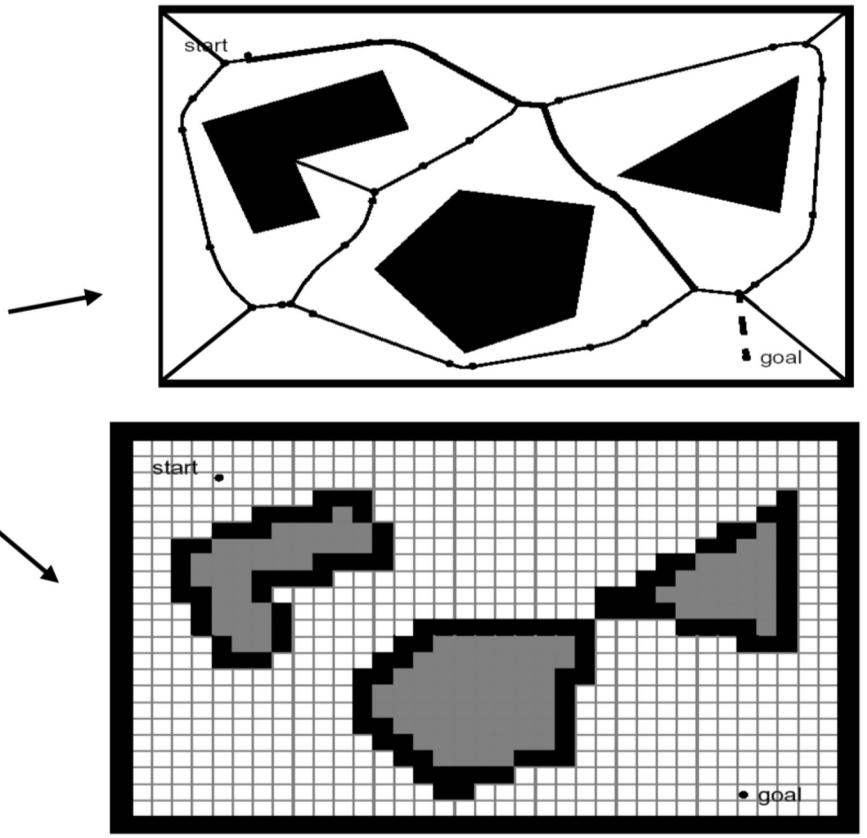
Fig. 1. Dashed blue spline is global path: a) Green spline is ideal local path; b) Red spline is actual local path

https://www.semanticscholar.org/paper/Generalized-Path-Corridor-based-Local-Path-Planning-Wang-Wieghardt/e97fea528a77be550e1262fbffa168e066198a44/figure/0

Different Approaches to Path Planning

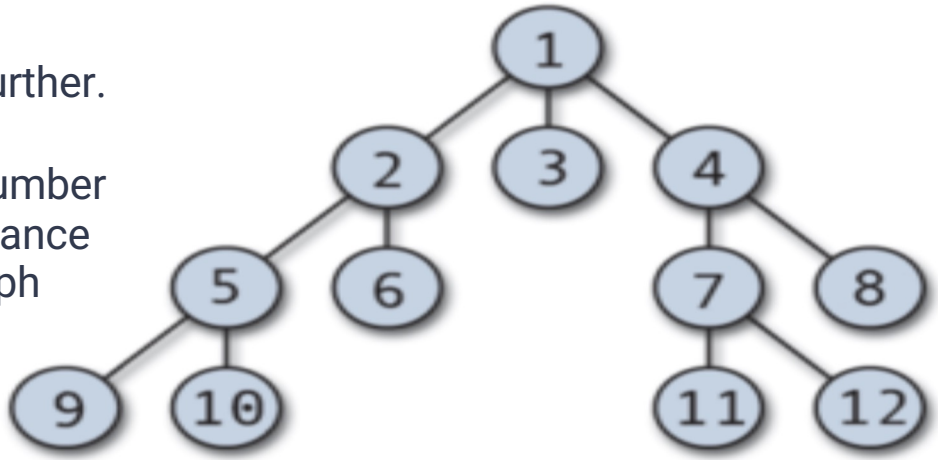
Capture the connectivity of free space into a graph that is subsequently searched for paths:

- *Road-map*: identify a set of routes within the free space
- *Cell decomposition*: discriminate between free and occupied cells -> connectivity graph



Graph Search Strategies

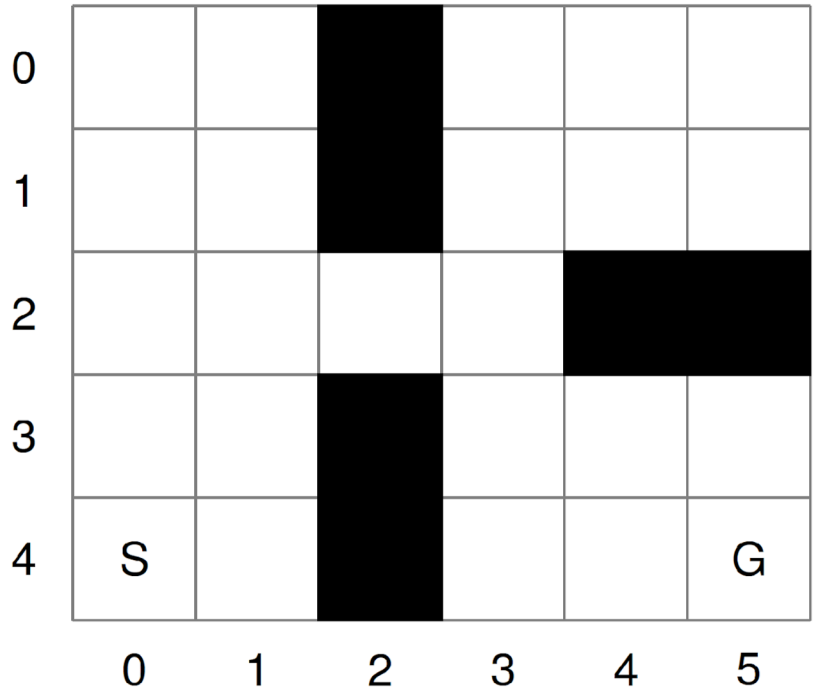
Breadth-first search (BFS)
• Explore all the nearest nodes before going one step further.
• Limit the risk of exploring deep dead-ends.
• The search always returns the path with the fewest number of edges between the start and the goal (minimum distance path assuming a constant cost of each edge in the graph



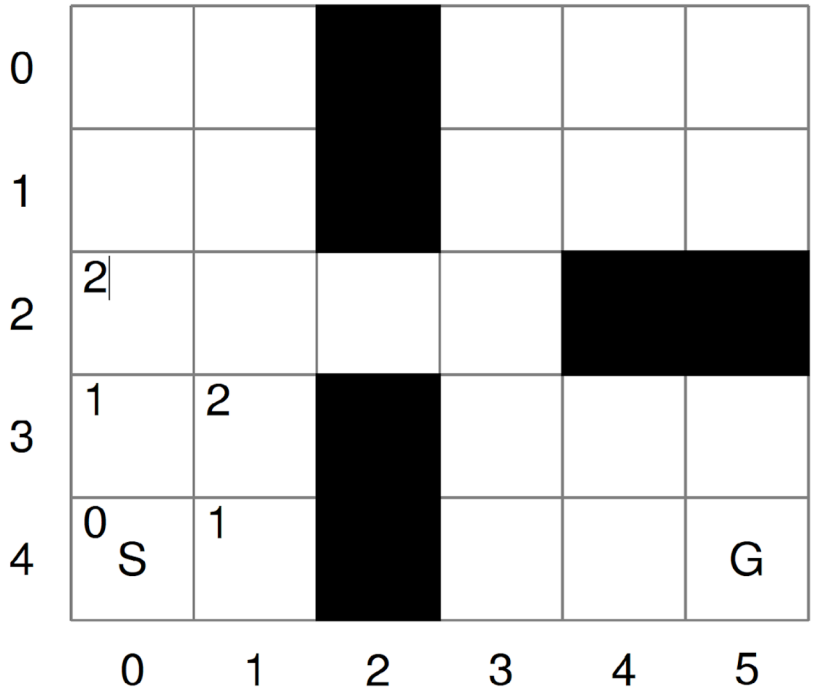
Depth-first search
• Explores as far as possible along each branch before backtracking.
• Memory efficient as completely explored branches may be deleted.

Best-first search (e.g. A*)
• Optimizes search by expanding the most promising node chosen according to some rule.
• Typically used for route finding: the straight-line distance, which is usually an approximation of road distance

Breadth-first example - Dijkstra's Algorithm, constant cost

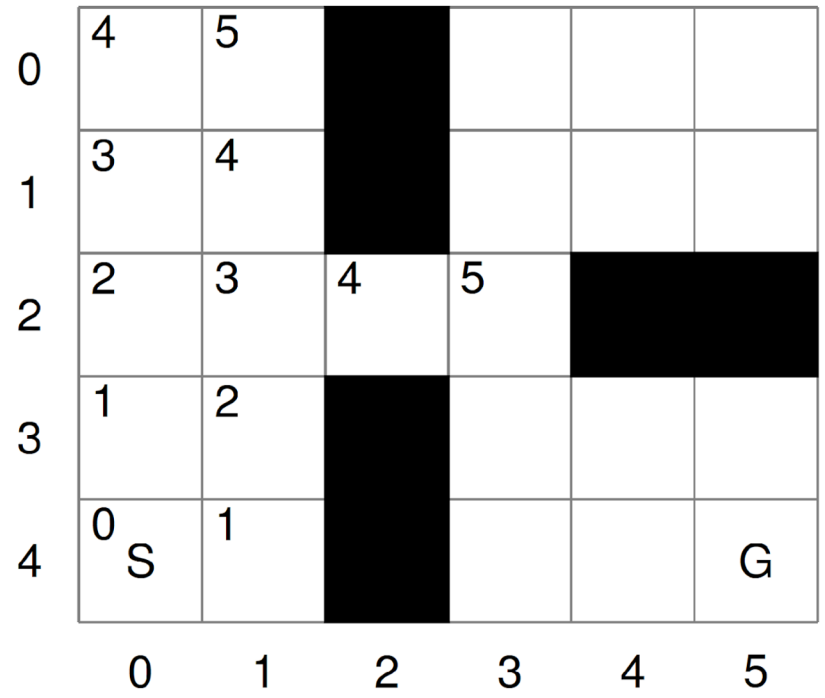


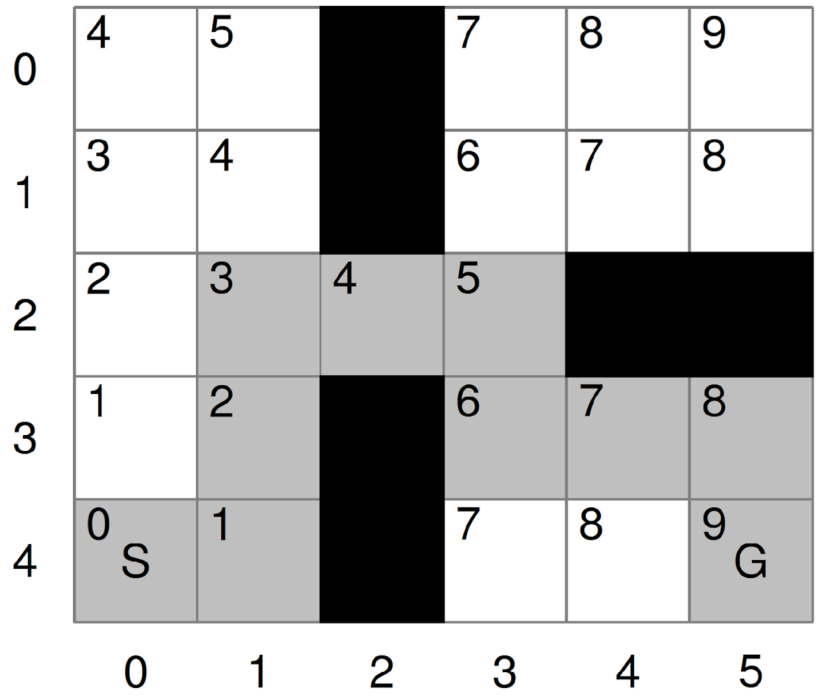
Breadth-first search



Manhattan distance

Breadth-first Example - Dijkstra's Algorithm, constant cost





Breadth-first Example - Dijkstra's Algorithm, constant cost

Algorithm 10.1: Dijkstra's algorithm on a grid map

```
integer n ← 0                          // Distance from start
cell array grid ← all unmarked         // Grid map
cell list path ← empty                 // Shortest path
cell current                           // Current cell in path
cell c                                 // Index over cells
cell S ← ···                           // Source cell
cell G ← ···                           // Goal cell
```

```
 1: mark S with n
 2: while G is unmarked
 3:     n ← n + 1
 4:     for each unmarked cell c in grid
 5:             next to a marked cell
 6:         mark c with n
 7: current ← G
 8: append current to path
 9: while S not in path
10:     append lowest marked neighbor c
11:         of current to path
12:     current ← c
```

Dijkstra's Algorithm, Variable Cost

2	1	2	3	4	5	6	7	8	9	10
1	S	1	2	3	4	5	6	7	8	9
2	1	2	3	4	5	6	7	8	9	10
3	2	3	4	5	9	10	11	12	13	11
4	3	4	5	9	13	14	15	16	13	12
5	4	5		13	14	15	16	15	14	13
6	5	6		14	15	16		16	15	14
7	6	7		15	16			G	16	15
8	7	8		14	15	16				16
9	8	9		13	14	15	16			
10	9	10	11	12	13	14	15	16		

Cost = 4 in central cells

2	1	2	3	4	5	6	7	8	9	10
1	S	1	2	3	4	5	6	7	8	9
2	1	2	3	4	5	6	7	8	9	10
3	2	3	4	5	7	8	9	11	12	11
4	3	4	5	7	9	10	11	13	13	12
5	4	5		9	10	11	12	13		13
6	5	6		10	11	12	13			
7	6	7		11	12	13	G			
8	7	8		12	13					
9	8	9		13						
10	9	10	11	12	13					

Cost = 2 in central cells

Best–first search – A* : Extending Dijkstra

Best-first search (e.g. A*)

- Optimizes search by expanding the most promising node chosen according to a combination of rules:

  lowest f(n) = g(n)+h(n)

  g(n) : motion cost (like in Dijkstra's algorithm)

  h(n) : heuristic function (e.g. distance to goal)

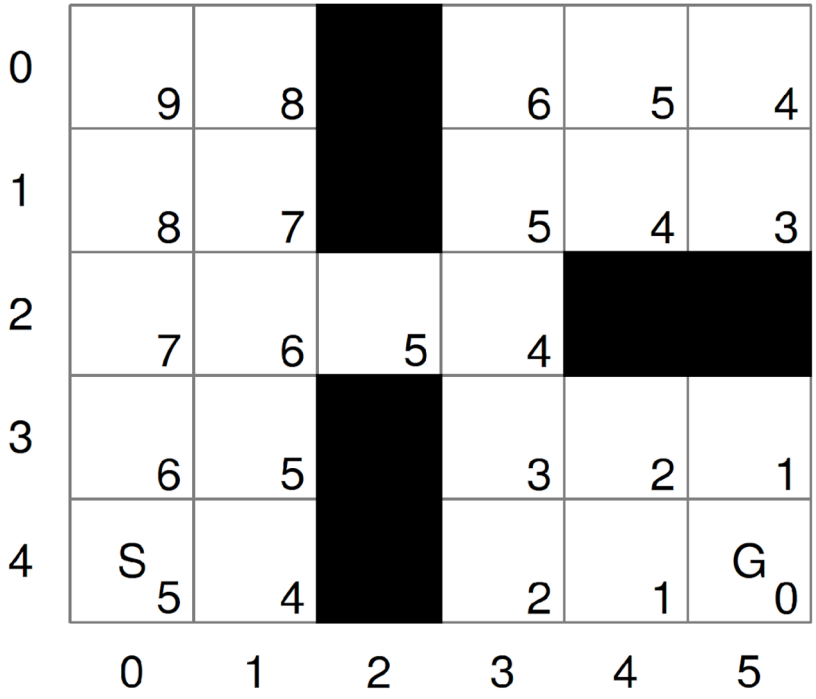
- Heuristic typically used for route finding: the straight-line distance, which is usually an approximation of road distance.

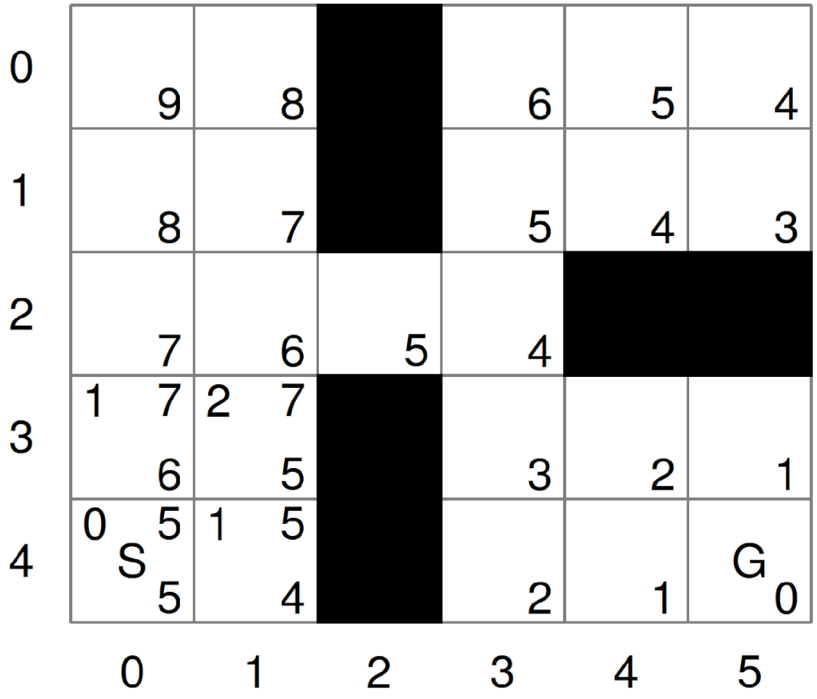
Best-first search - A* : Extending Dijkstra

h(n) : heuristic function (d to goal)
g(n) : motion cost (like Dijkstra)

f(n) = g(n)+h(n)

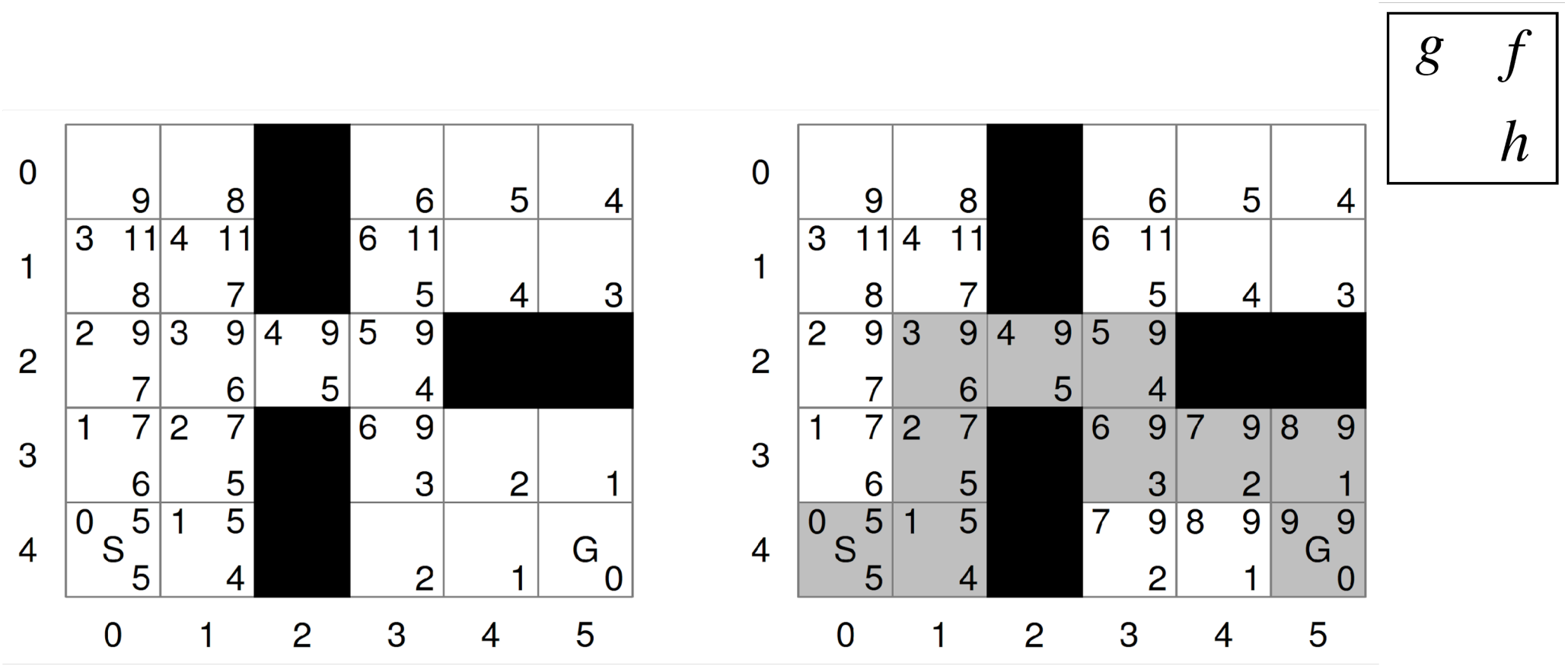
g f
h





h(n) is simple to compute

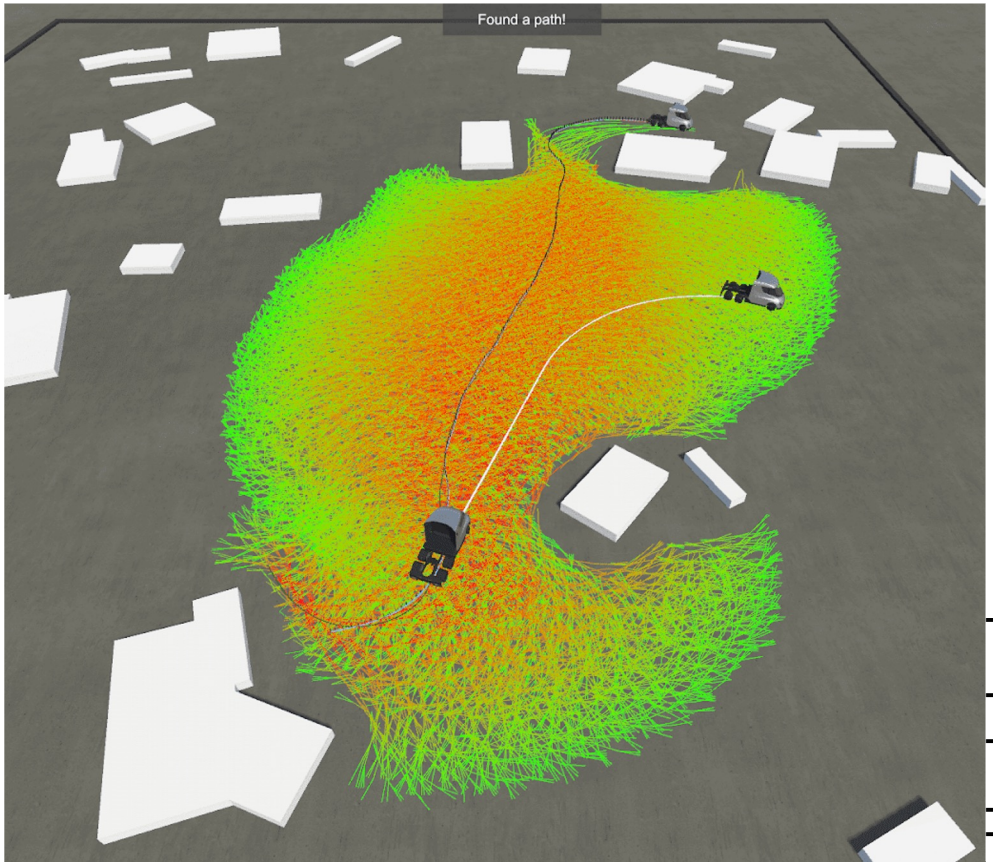
Best-first search - A* : Extending Dijkstra



Closer to reality: Hybrid A*

You can add constraints to the choice of cells, for instance

- add a cost when turning
- limit turning angle and access only to some cells, fitting to the possibility of your vehicle (car)
- differentiate forward and backwards motion

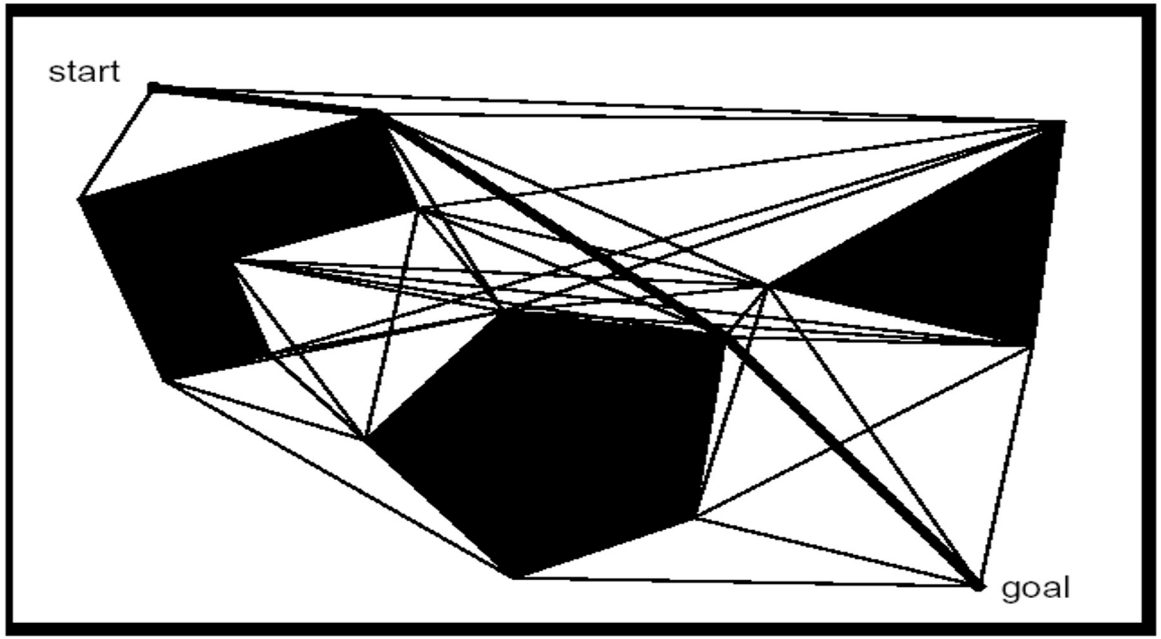


blog.habrador.com

Approach 1 Finding the road – Visibility Graphs

The challenge is to construct a set of roads that enable the robot to go from start to goal, while minimizing the number of total roads.

Joining all pairs of vertices that can see each other (including initial and goal position).



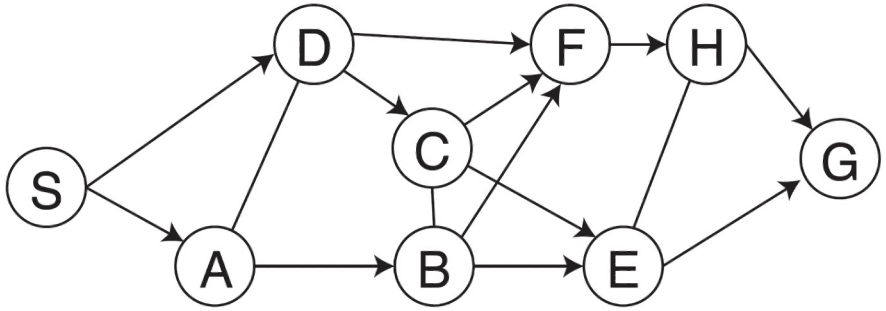
DEMO

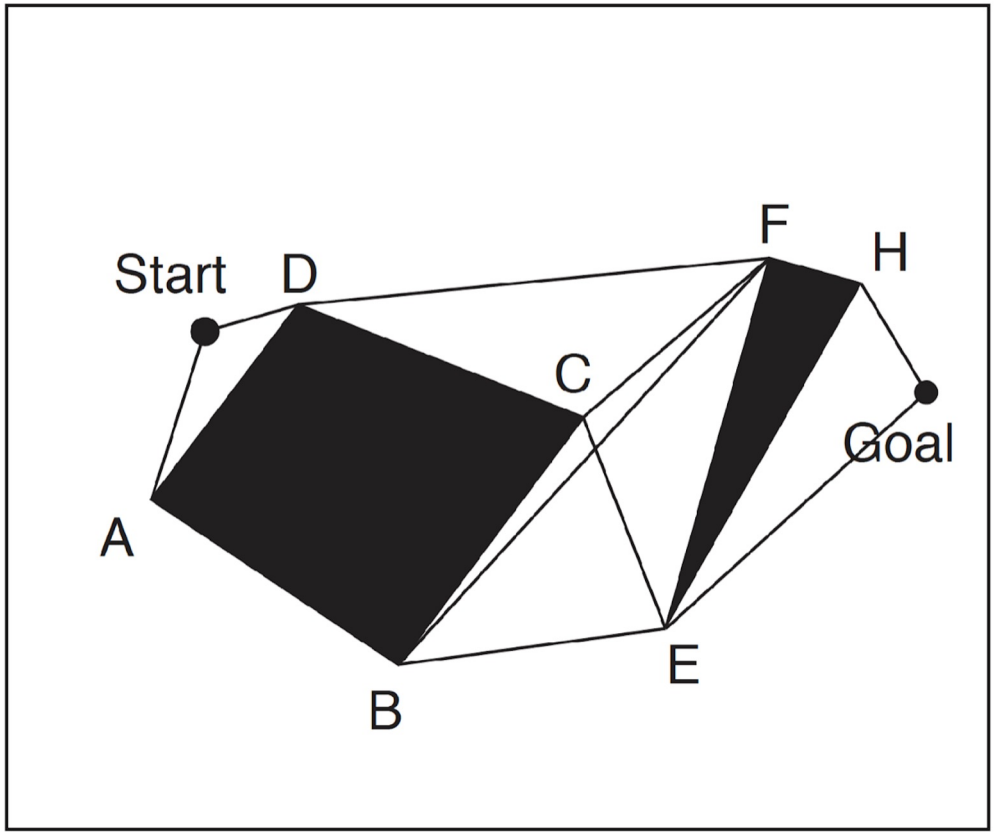
Approach 1 Finding the road – Visibility Graphs

Search for shortest path length using a graph search technique.

+ Complete.

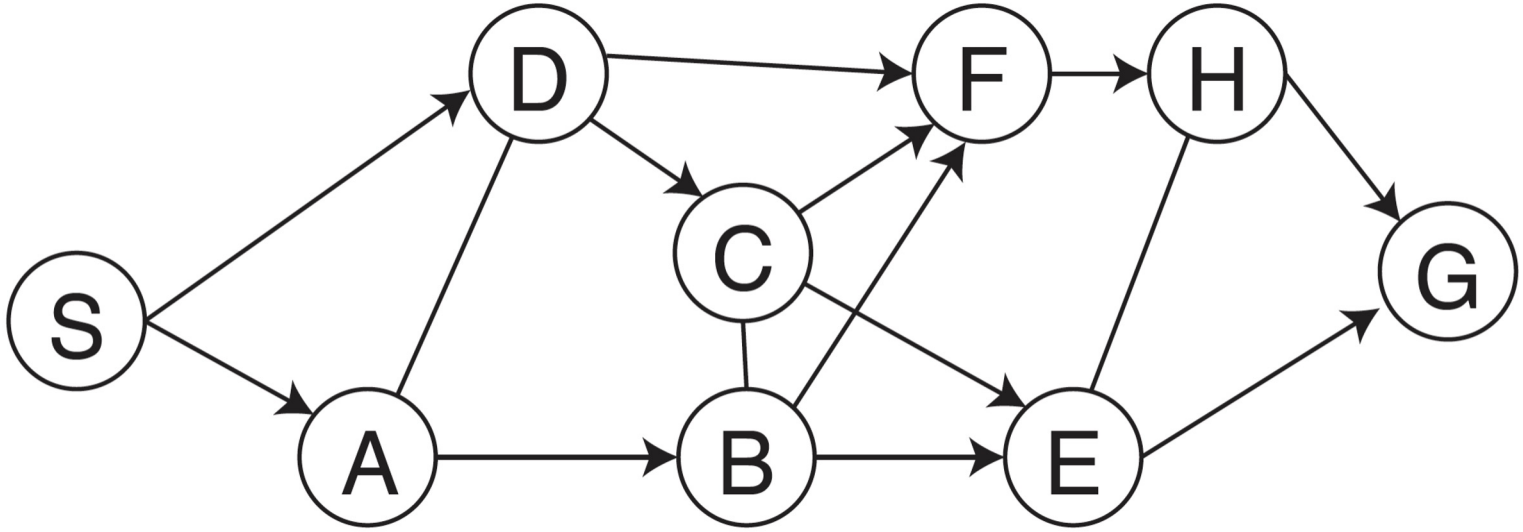
- Tend to graze obstacles
=> requires to grow obstacles to avoid collisions.





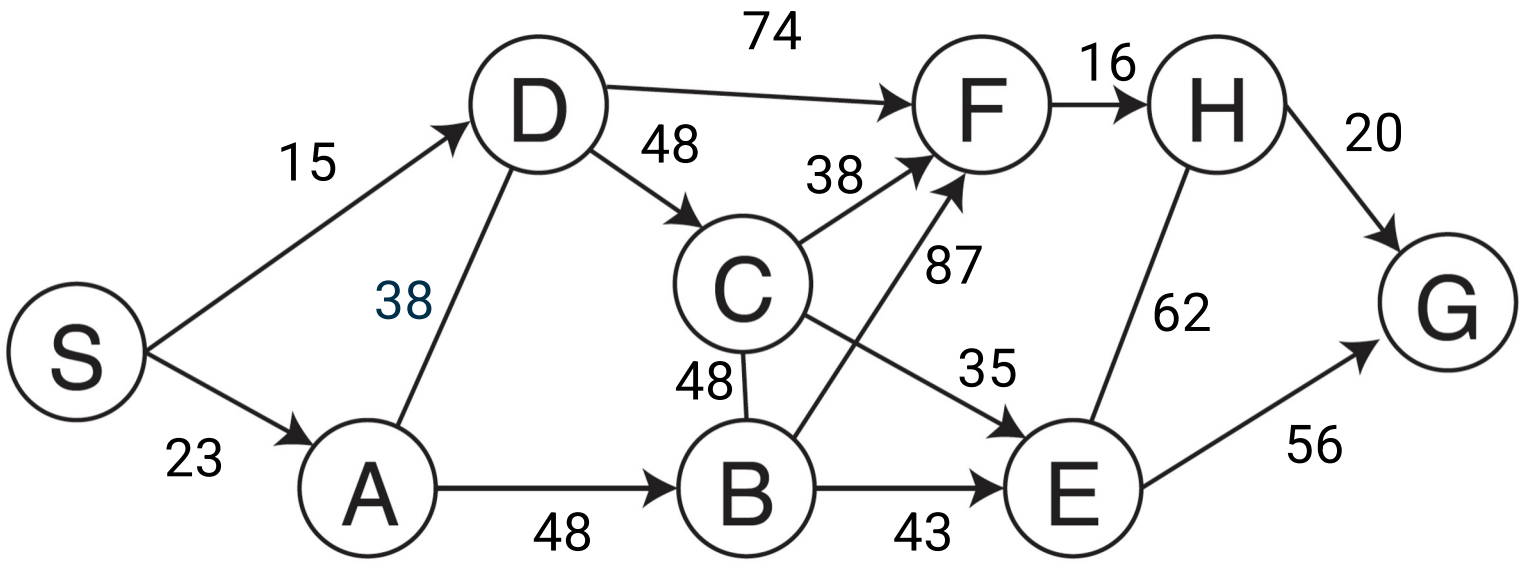
Approach 1 Finding the road – Visibility Graphs

Search for shortest path length using Dijkstra's Algorithm with real distance.



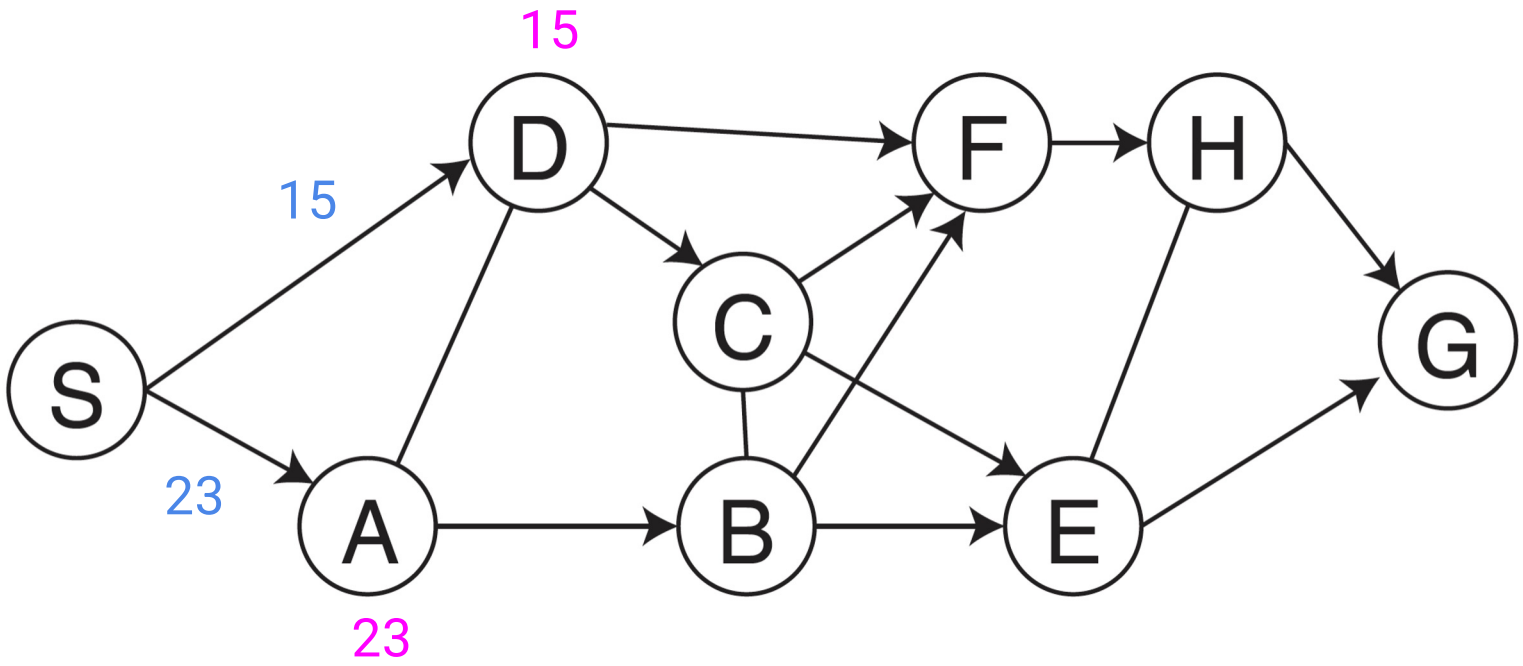
Approach 1 Finding the road – Visibility Graphs

Associate a distance with each segment of the graph



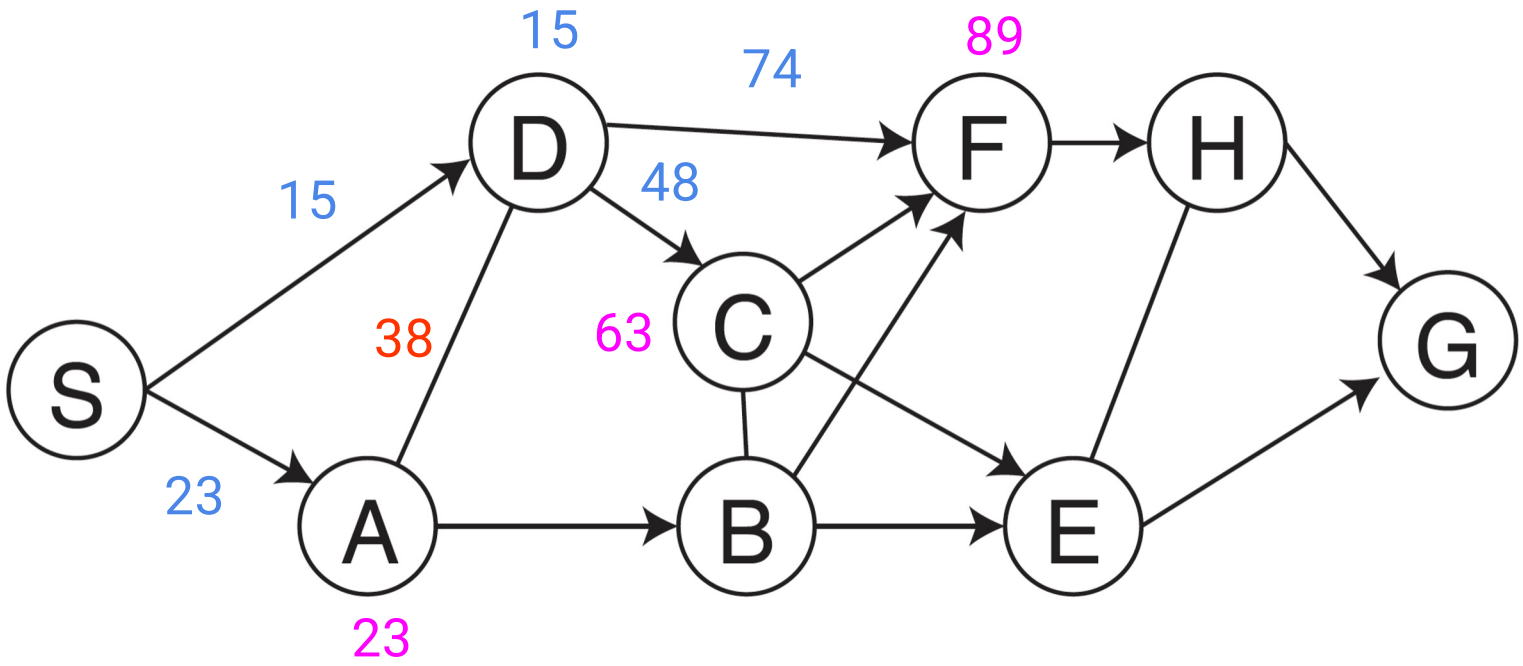
Approach 1 Finding the road – Visibility Graphs

Search for shortest path length using Dijkstra's Algorithm with real distance.



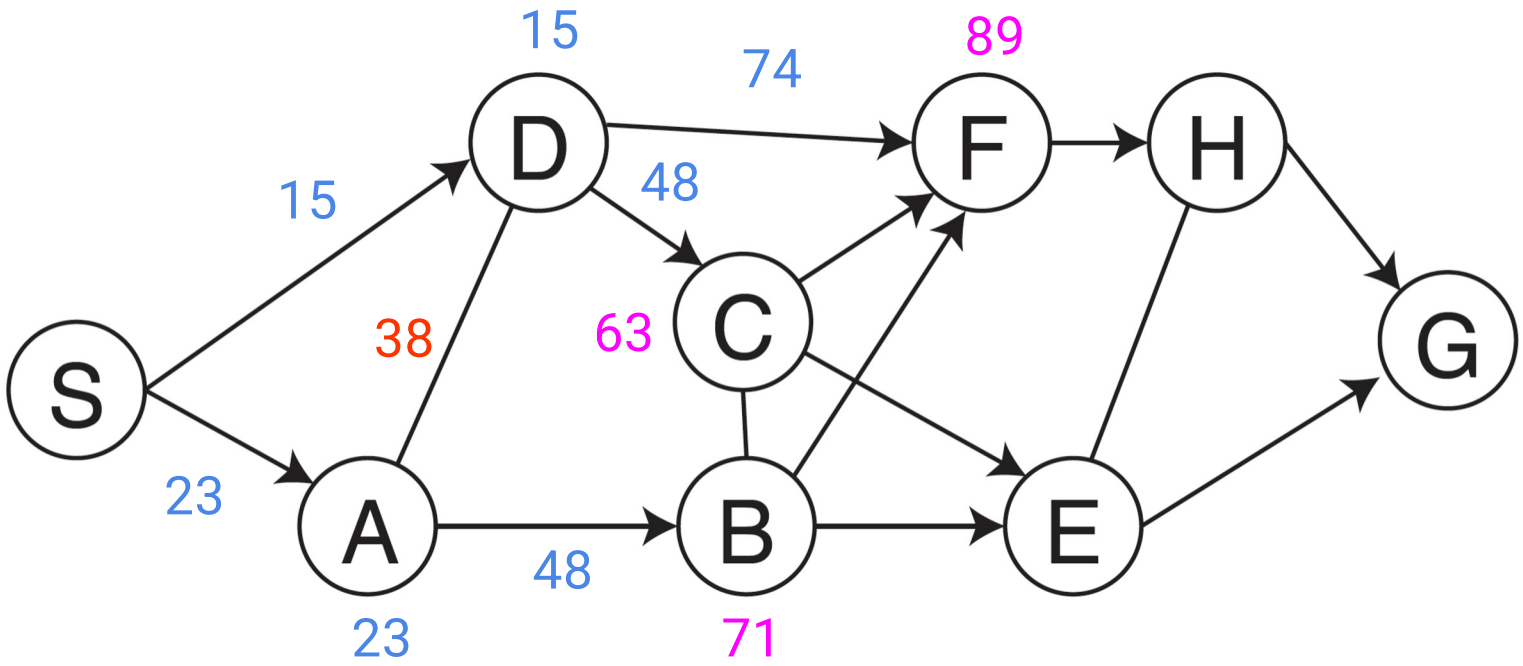
Approach 1 Finding the road – Visibility Graphs

Search for shortest path length using Dijkstra's Algorithm with real distance.



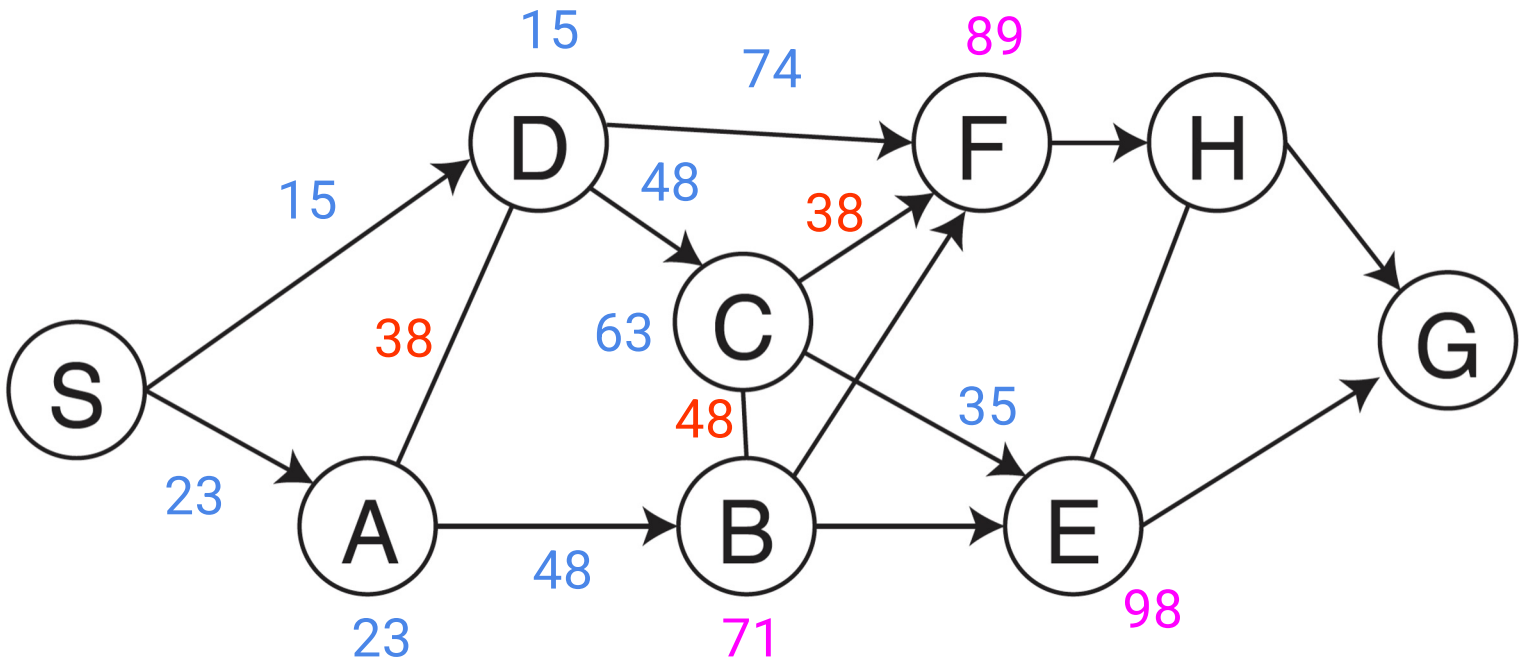
Approach 1 Finding the road – Visibility Graphs

Search for shortest path length using Dijkstra's Algorithm with real distance.



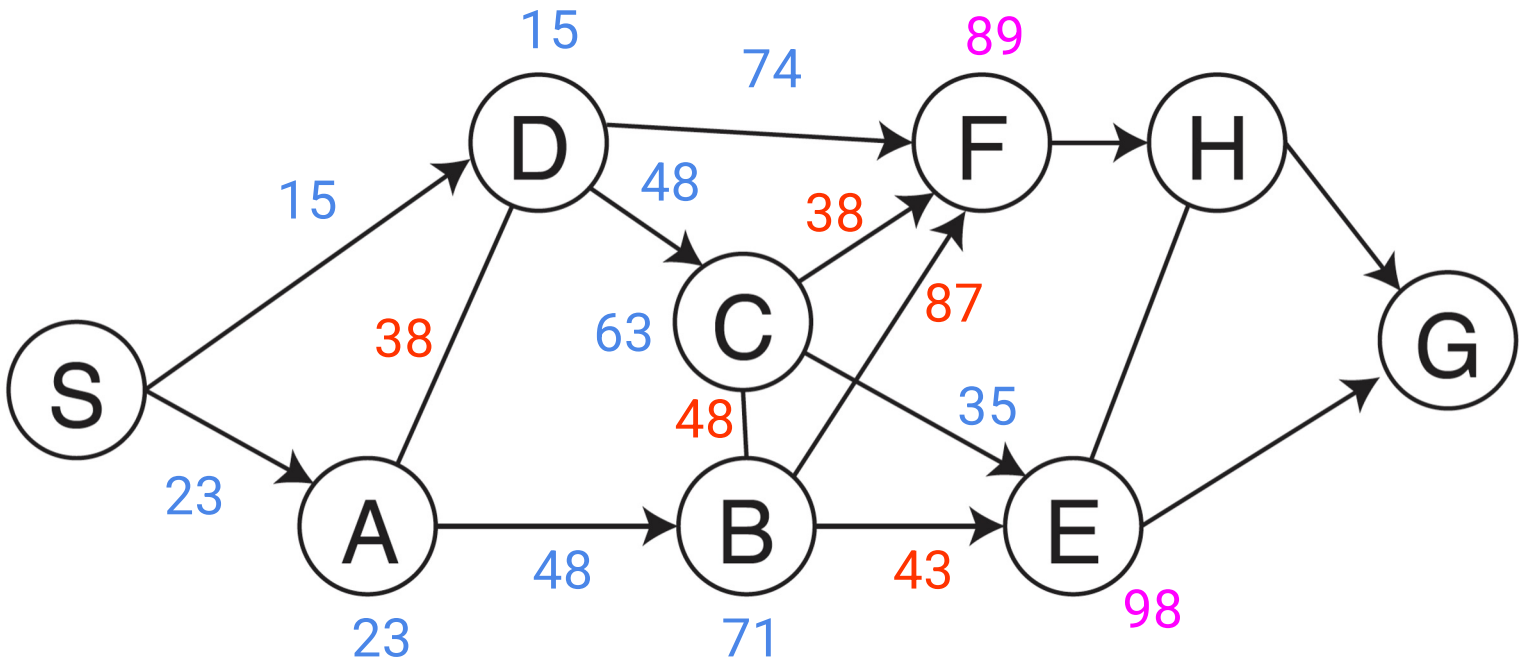
Approach 1 Finding the road – Visibility Graphs

Search for shortest path length using Dijkstra's Algorithm with real distance.



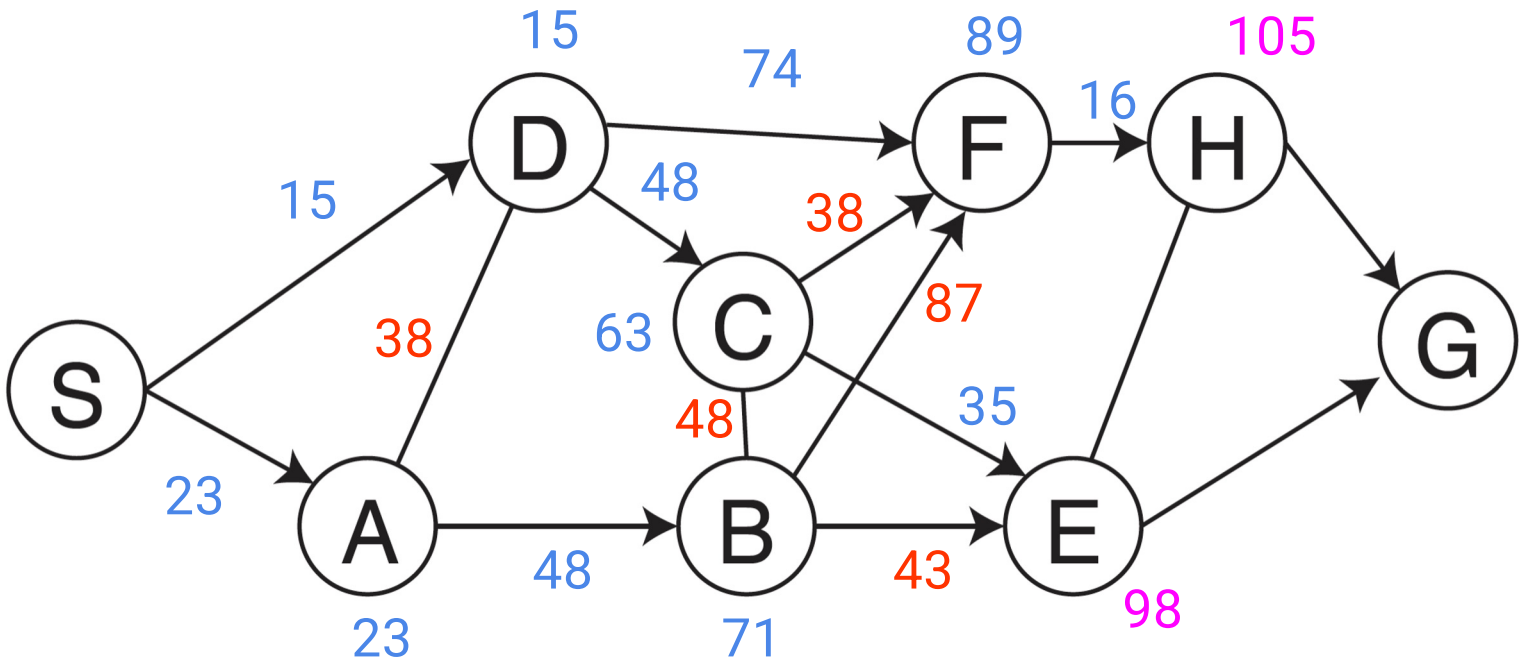
Approach 1 Finding the road – Visibility Graphs

Search for shortest path length using Dijkstra's Algorithm with real distance.



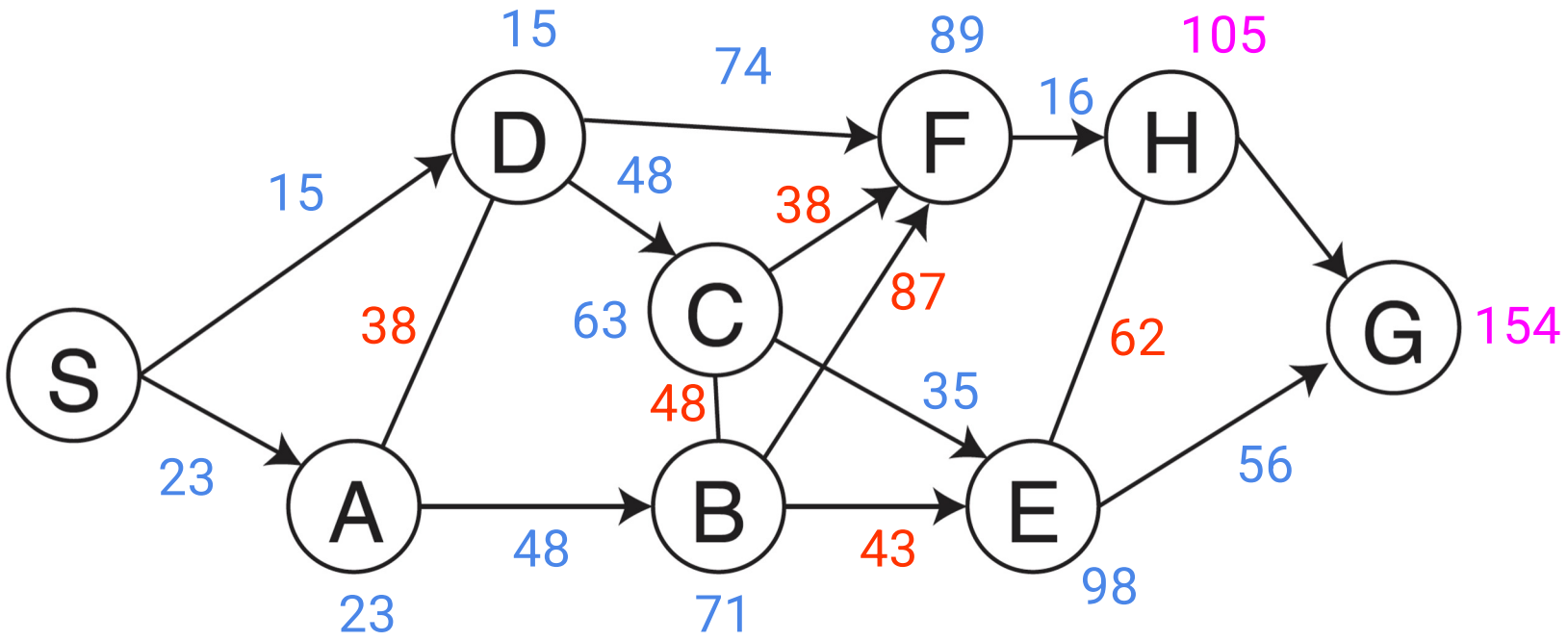
Approach 1 Finding the road – Visibility Graphs

Search for shortest path length using Dijkstra's Algorithm with real distance.



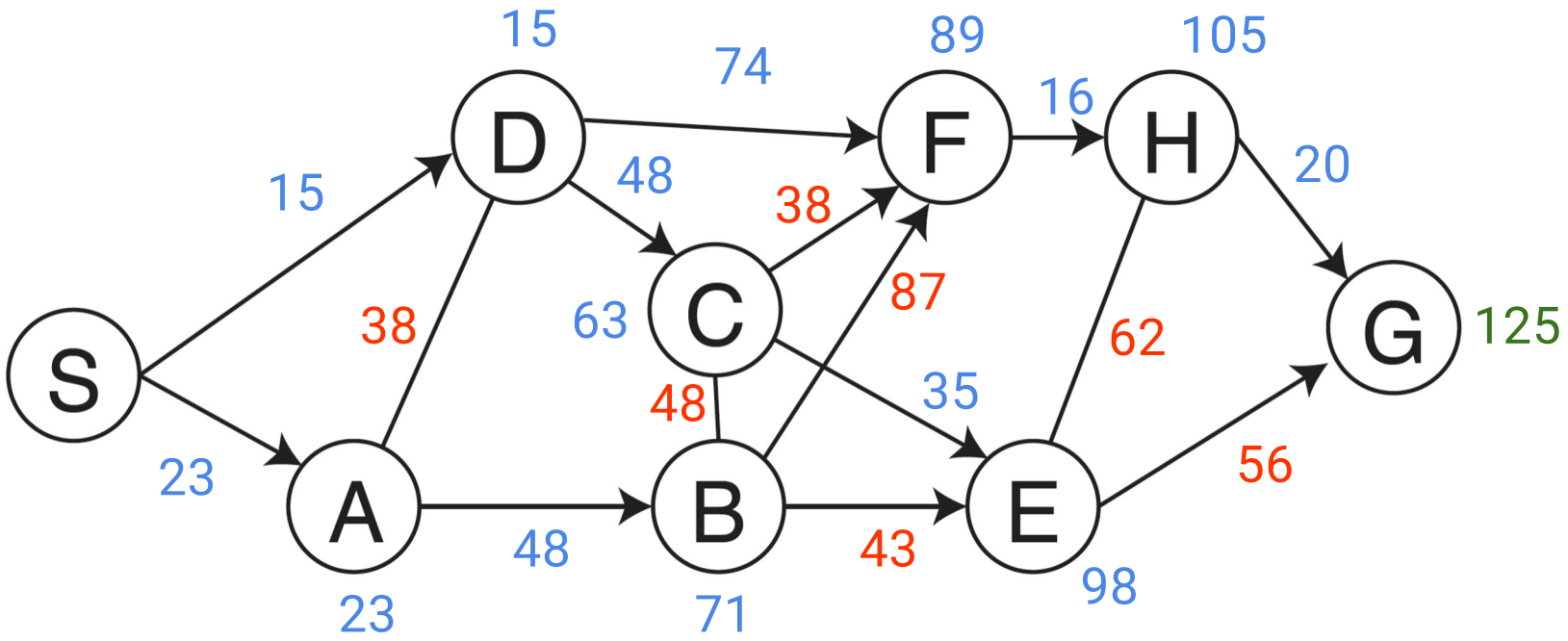
Approach 1 Finding the road – Visibility Graphs

Search for shortest path length using Dijkstra's Algorithm with real distance.



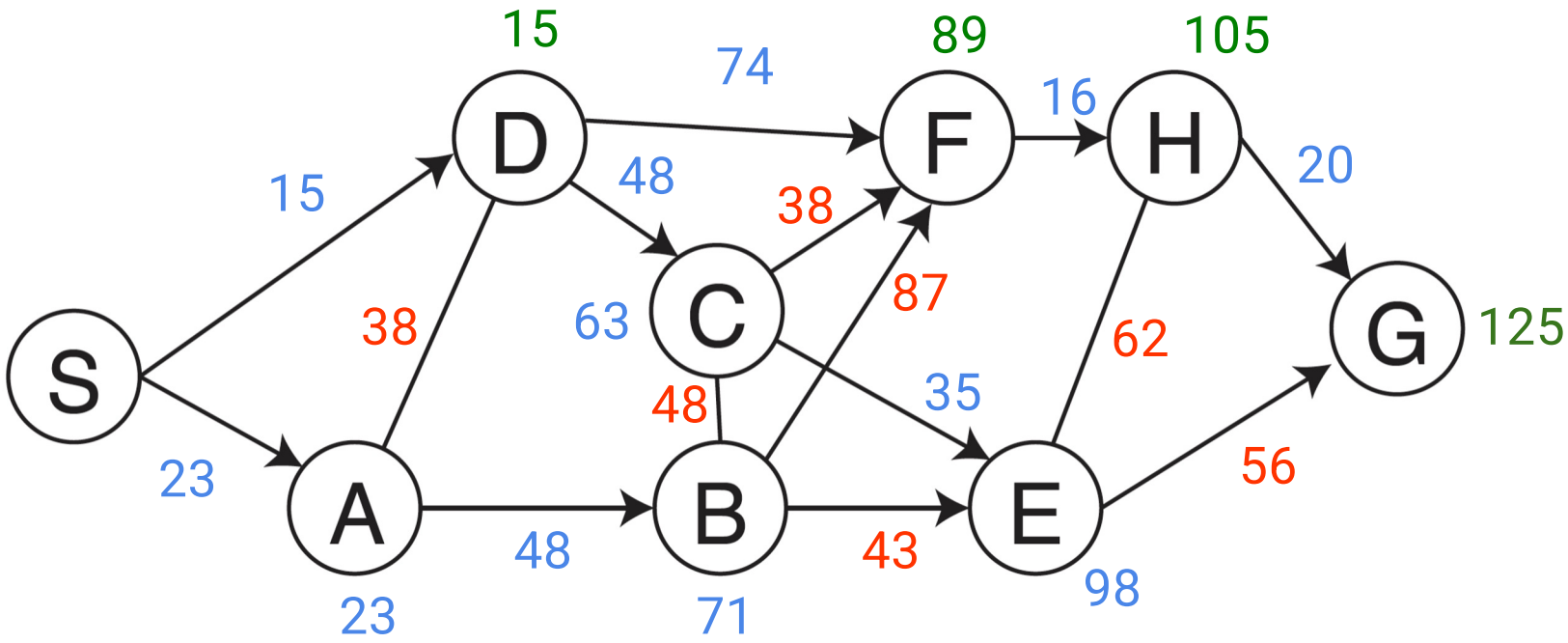
Approach 1 Finding the road – Visibility Graphs

Search for shortest path length using Dijkstra's Algorithm with real distance.



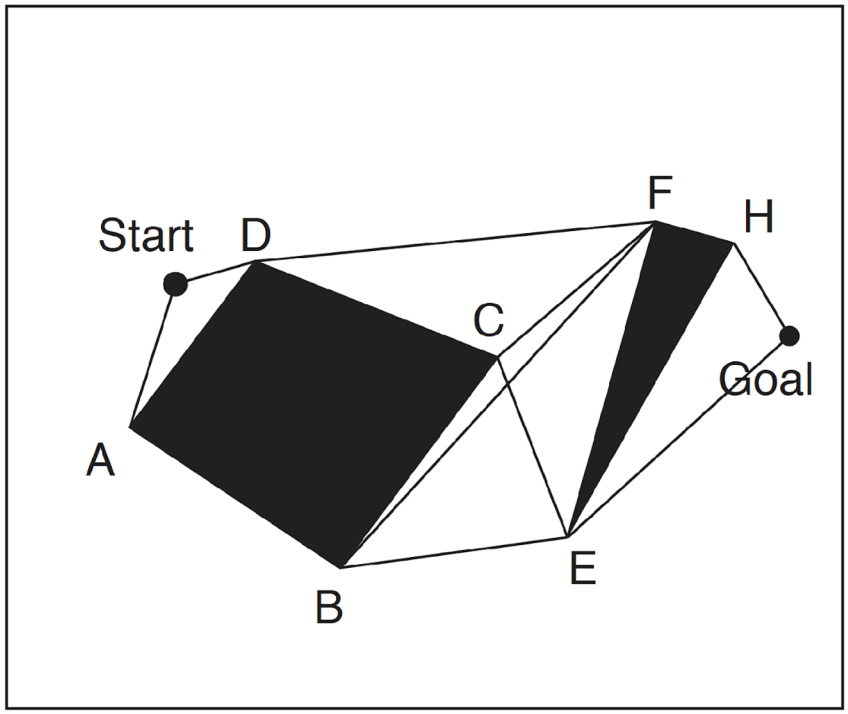
Approach 1 Finding the road – Visibility Graphs

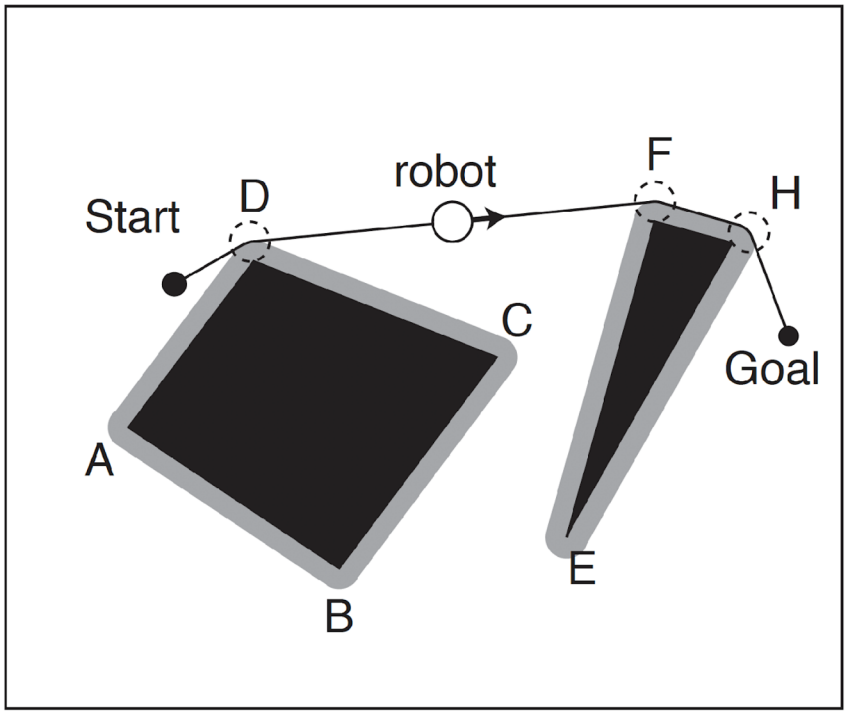
Search for shortest path length using Dijkstra's Algorithm with real distance.



Approach 1 Example – Visibility Graphs

Requires growing obstacles to avoid collisions.





= robot size

Approach 1 Example – Voronoi Diagrams

For each point in the free space compute its distance from the nearest obstacle (growing circles) => results in sharp ridges.

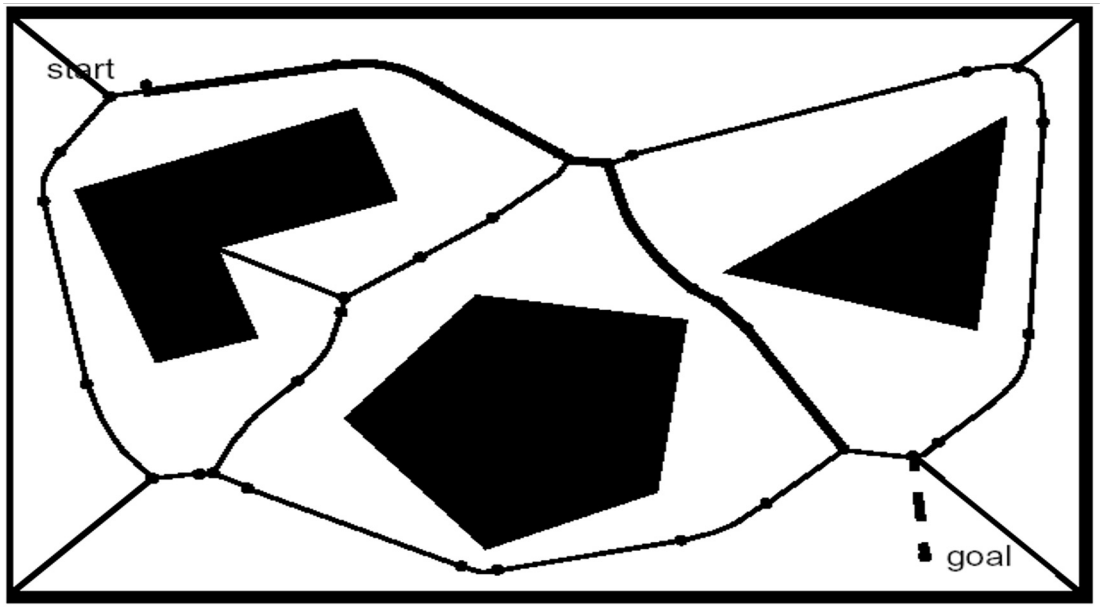
When the configuration space obstacles are polygons, the Voronoi diagram consists of straight and parabolic segments.

Tends to maximize the clearance between the robot and the obstacles.

+ May be straight forward to execute when using range scanner (the robot maximizes the readings of local minima in its sensor values).

- Far from optimal in the sense of total path length.

- May be problematic for localization with short range sensors since no obstacle may be sensed most of the time.



Approach 2 – Cell Decomposition

Divide free space into simple, connected regions called cells.

Determine which open cells are adjacent and construct a *connectivity graph*.

Find cells in which the initial and goal positions lie and search for a path in the connectivity graph to join them.

From the sequence of cells found with an appropriate search algorithm, **compute a trajectory within each cell**, e.g.

- passing through the midpoints of cell boundaries or
- by sequence of line following movements.

Two families of cell decomposition methods:

- *exact*: cells are either completely free or completely occupied;
- *approximate*: not taking care of the obstacle geometry.

Approach 2 Example – Exact Cell Decomp.

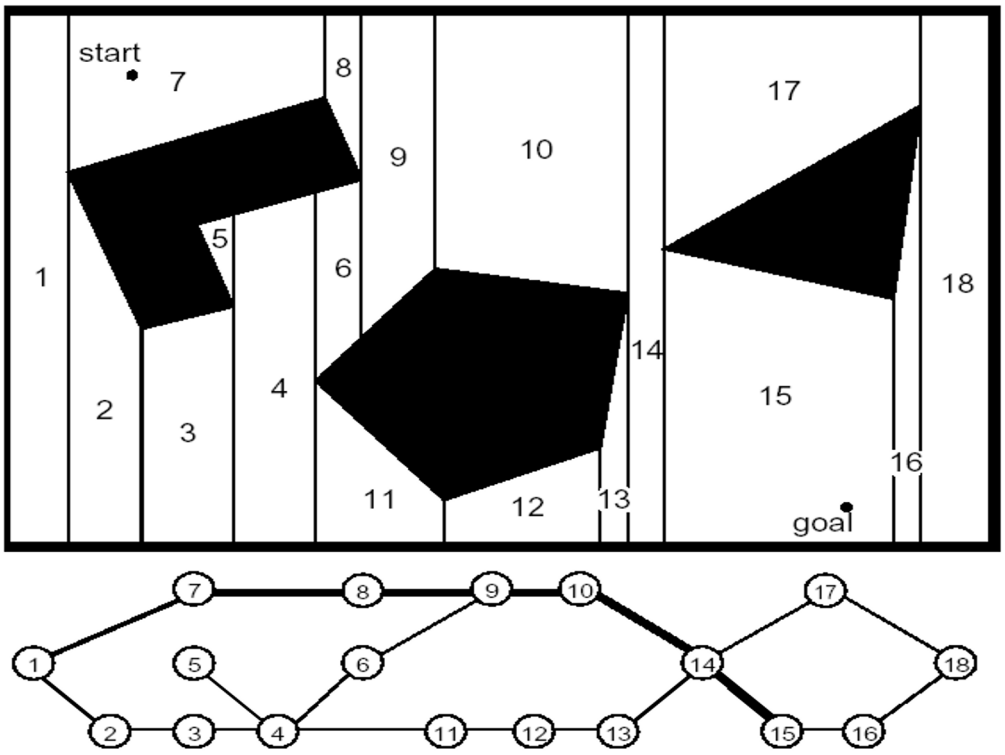
The boundary of the cells is based on geometric criticality, for example the edges of the obstacles.

Underlying assumption: the particular position of the robot within each cell of free space does not matter.

+ Number of cells depend on density and complexity of objects in the environment.

+ Complete.

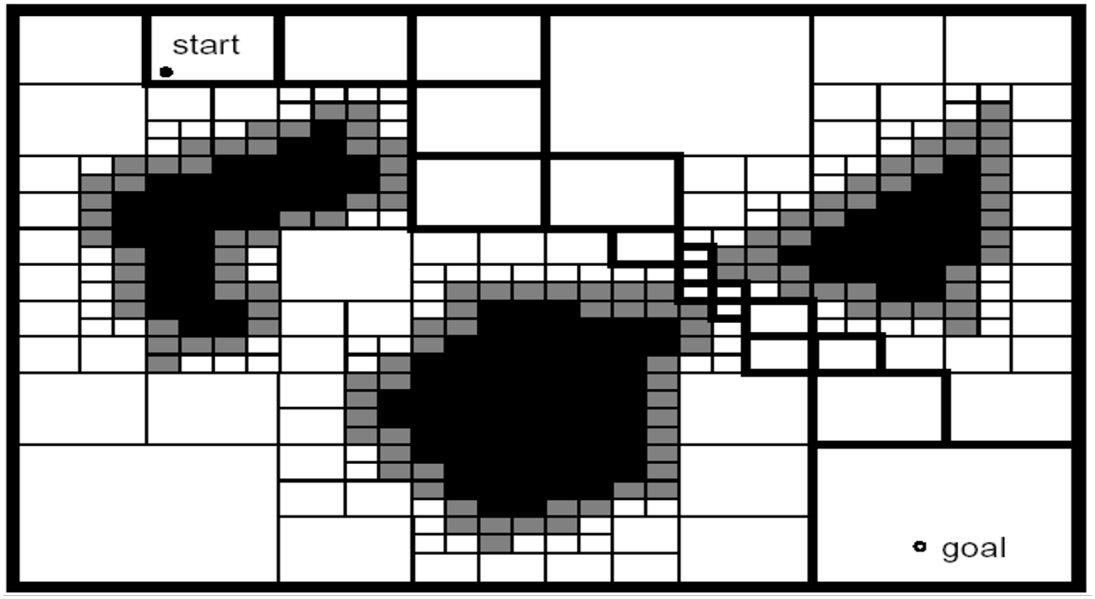
- Dependent on the density and complexity of obstacles.



Approach 2 Example – Approx Cell Decomp. 1

First example: adaptive cell decomposition

- The rectangle is decomposed into 4 identical rectangles.
- If the interior of a rectangle lies completely in free space or on an obstacle, it is not further decomposed. Otherwise, it is recursively decomposed into 4 rectangles, and so on. Stop at a given minimal cell size.



+ Adapted to the complexity of the environment.

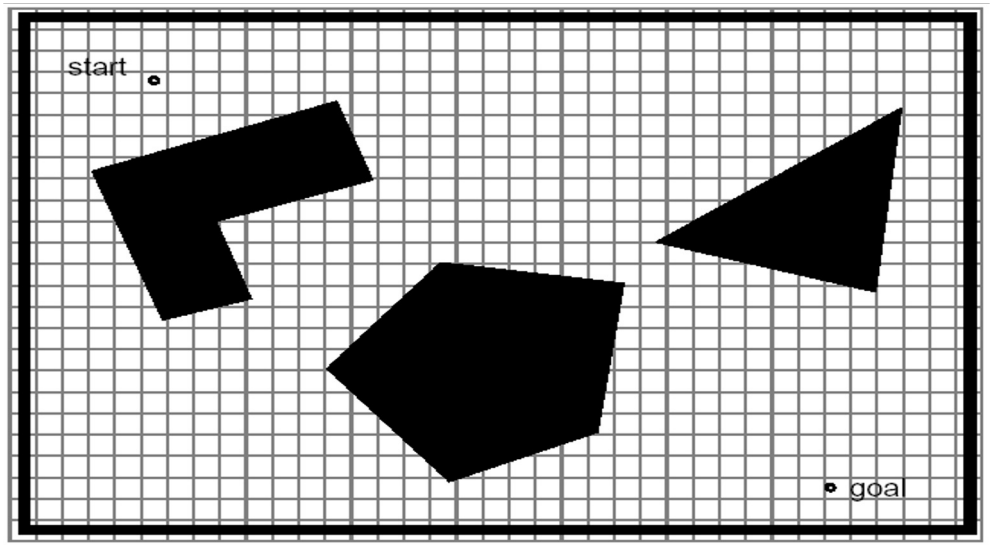
- May not be complete (depending on the minimal cell size).

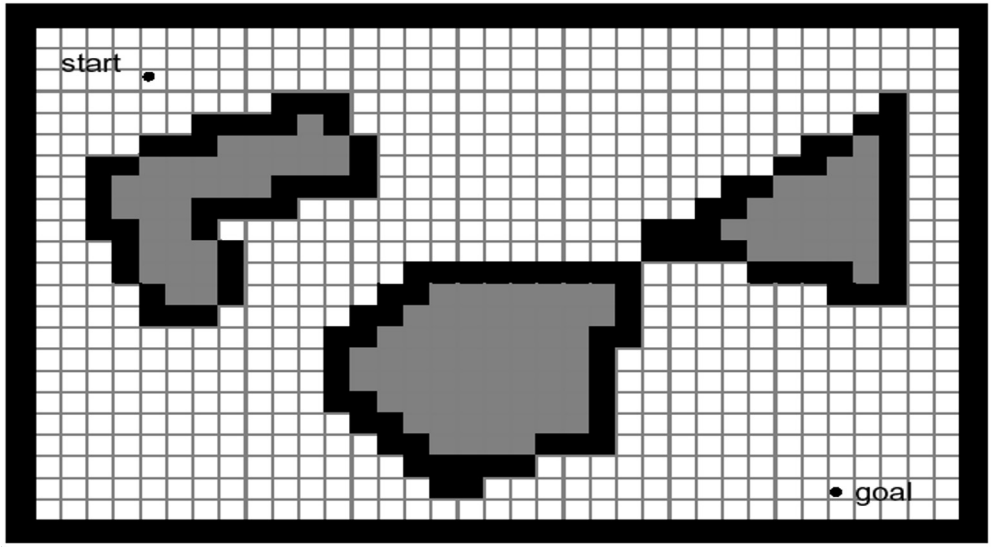
Approach 2 Example – Approx Cell Decomp. 2

2nd example: fixed grid-size decomposition

- Narrow passageways can be lost => may not be complete.

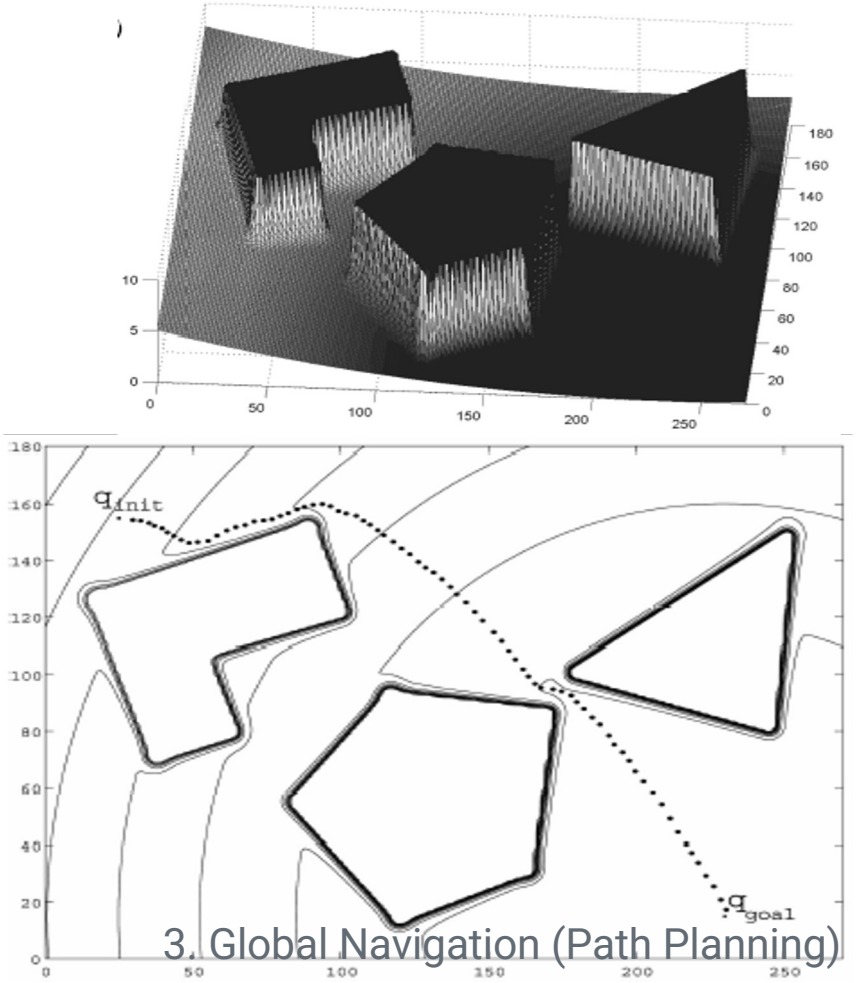
+ Very simple path planning algorithm such as wavefront expansion can be used => computationally efficient.





Approach 3 Example – No graph, Potential Field

- Robot is treated as a particle under the influence of an artificial potential field:
  - Goal generates attractive force.
  - Obstacles are repulsive forces.
- At every location, the direction of the resulting force is considered the most promising direction of motion.



Approach 3 Example – Potential Field Generation

Generation of potential field function *U(q)*, where *q* is the state variable:

- attracting (goal) and repulsing (obstacle) fields
- summing up the fields
- functions must be differentiable

Generate artificial force field *F(q)*

$$F(q) = -\nabla U(q) = -\nabla U_{att}(q) - \nabla U_{rep}(q) = \begin{bmatrix} \frac{\partial U}{\partial x} \\ \frac{\partial U}{\partial y} \end{bmatrix}$$

Approach 3 Example – Potential Field Generation

If the *robot is holonomic*, the control can be: set robot speed (v_x,v_y) proportional to the force $F(q)$ generated by the field (assuming that the dynamics is negligible).

If the *robot is nonholonomic*, a transformation needs to be found. For instance, with a differential-drive, the rotation speed can be set proportional to the angle α between $F(q)$ and the current orientation of the robot, whereas the forward speed can be set proportional to the amplitude of $F(q)$ or $F(q)$•cos(α) (in order to avoid moving too fast forward if the current orientation of the robot is not aligned with $F(q)$).

Approach 3 Example – Notes on Potential Field

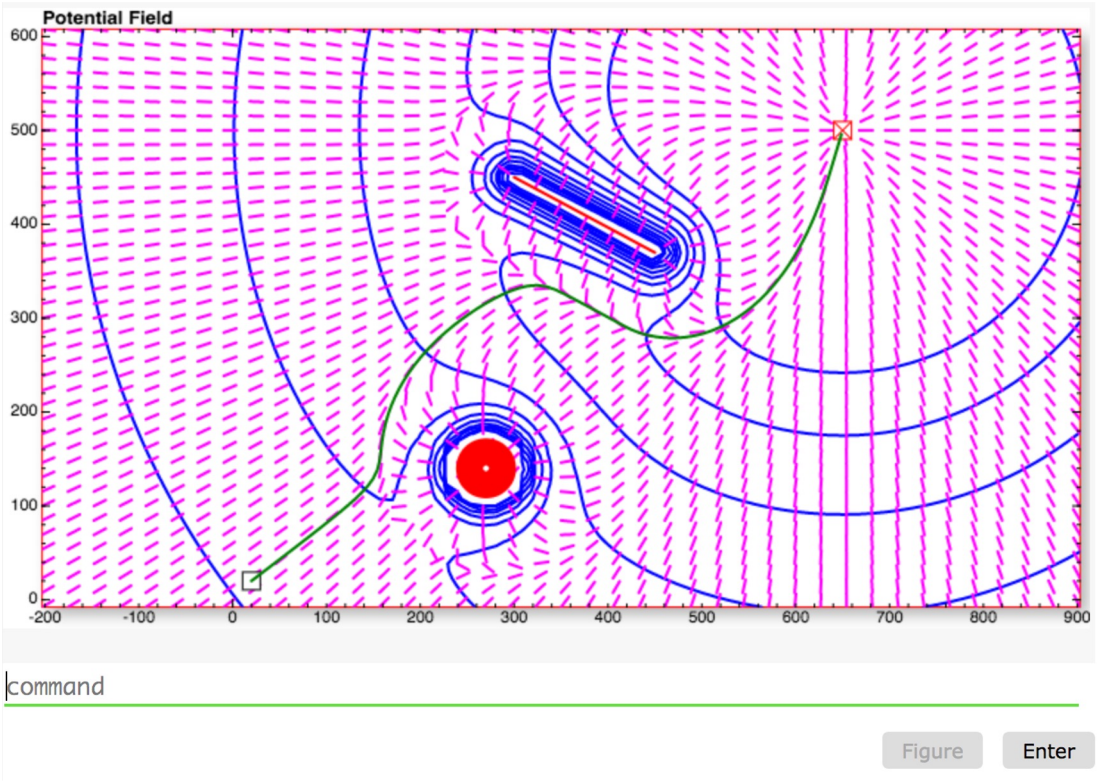
- The potential field methods are usually incomplete.
- Local minima problem exists → can result in oscillations or dead-locks.
- Problem is getting more complex if the robot cannot be considered as a point mass.

+ Easy to implement efficient and reasonably reliable planners.

+ Can be used off-line to draw a complete path that is then followed using a low level controller.

+ Can be used on-line to compute only the force present at current pose and control the robot accordingly.

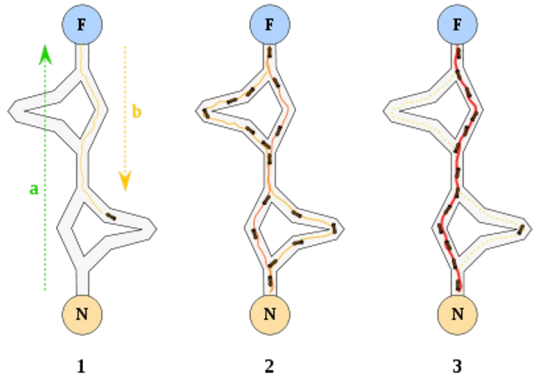
Demo 3 on Potential Field



Stigmergy Based Path Optimisation

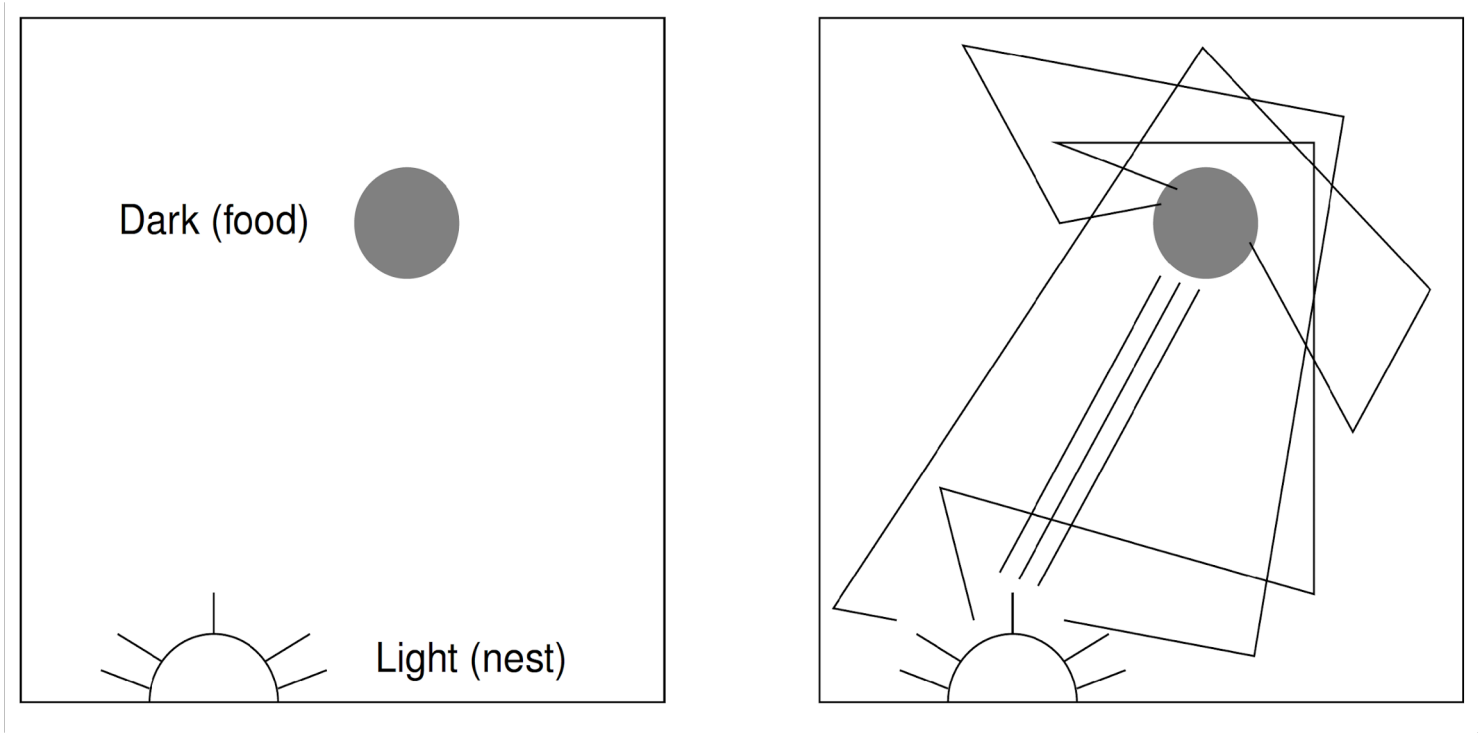
- Create the plan directly in the environment, generating global path with local information

- Marking the environment (simulating pheromones in ants, for instance)
- Can be done by multiple robots or single robots
- A known version: Ant Colony optimisation

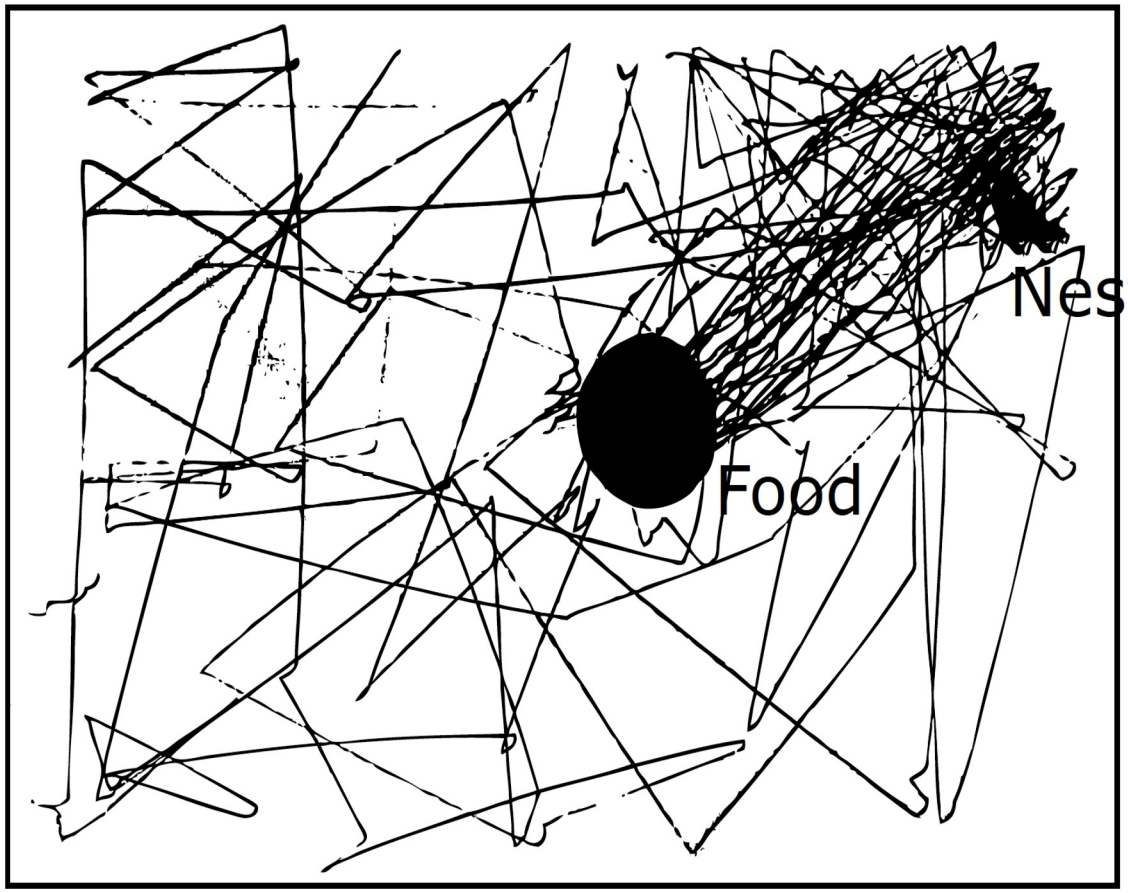


Wikipedia, ACO

Stigmergy Based Path Optimisation

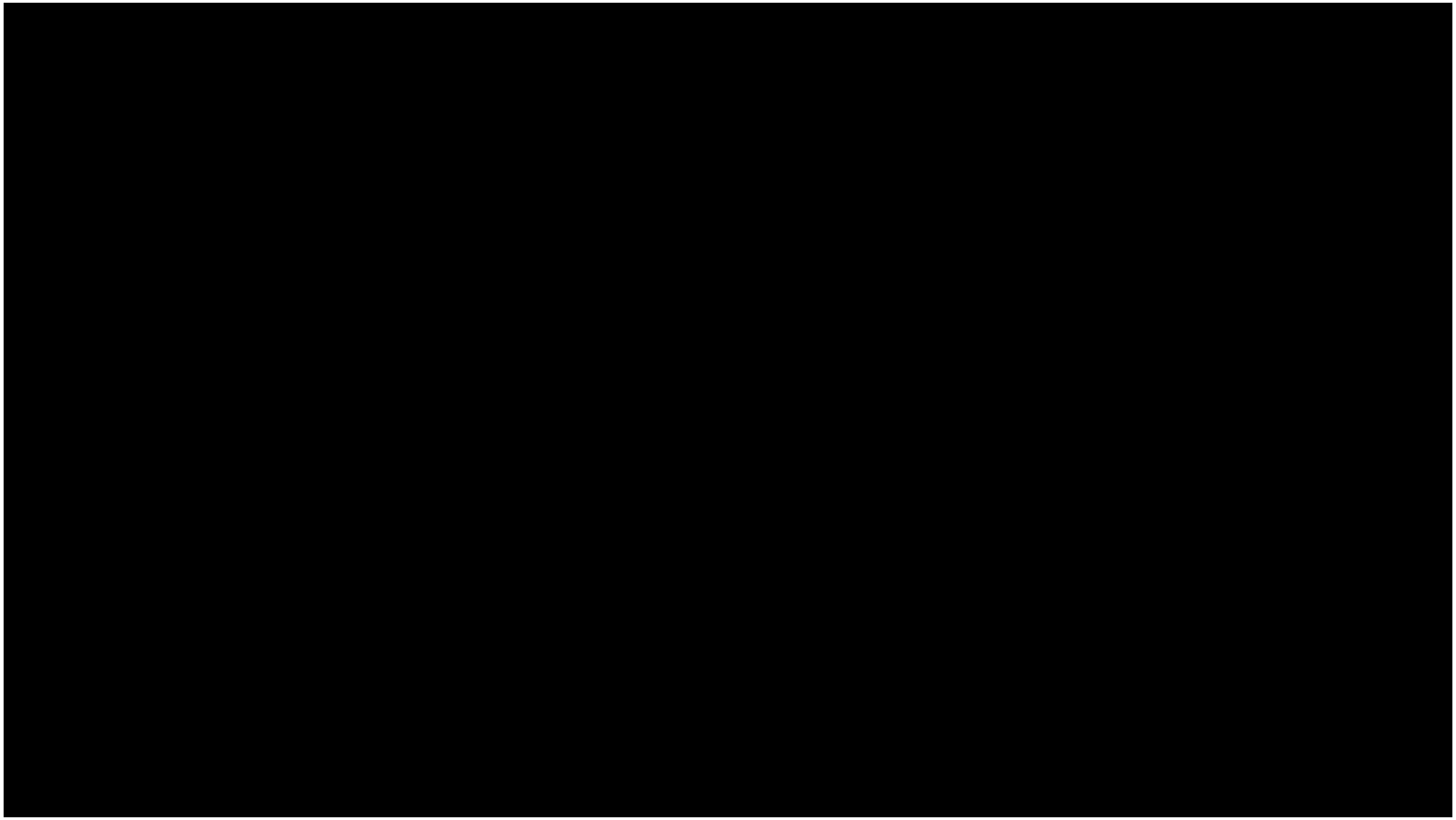


Stigmergy Based Path Optimisation



Stigmergy Based Path Optimisation

- White: looking for food
- Red: food found, back to nest depositing pheromone
- Green: following trail toward food



https://www.youtube.com/watch?v=V4Vlus5HWLA

Ant colony optimization

- Simulation of ants, extracting the optimal path
- Based on pheromone trail with evaporation (unlike previous example)
- Used for optimisation of complex path problem:
  - Vehicle routing
  - Computer network routing
  - Agent scheduling
  - ...

What should I remember

- Path planning
  - Definition
  - Road-based approaches vs space-based approaches
  - Graph search strategies
  - Algorithms for graph search:
    - Dijkstra, with constant and variable cost
    - A*
  - Algorithms for graph creation (approach, advantages and disadvantages):
    - visibility graph, including the obstacle increase by radius of robot
    - Voronoi Diagrams
    - Exact cell decomposition
    - Adaptive cell decomposition
    - Fixed grid-size decomposition
  - Potential field (approach for global nav, advantages and disadvantages)
  - Stigmergy Based Path Optimisation

NEXT WEEK

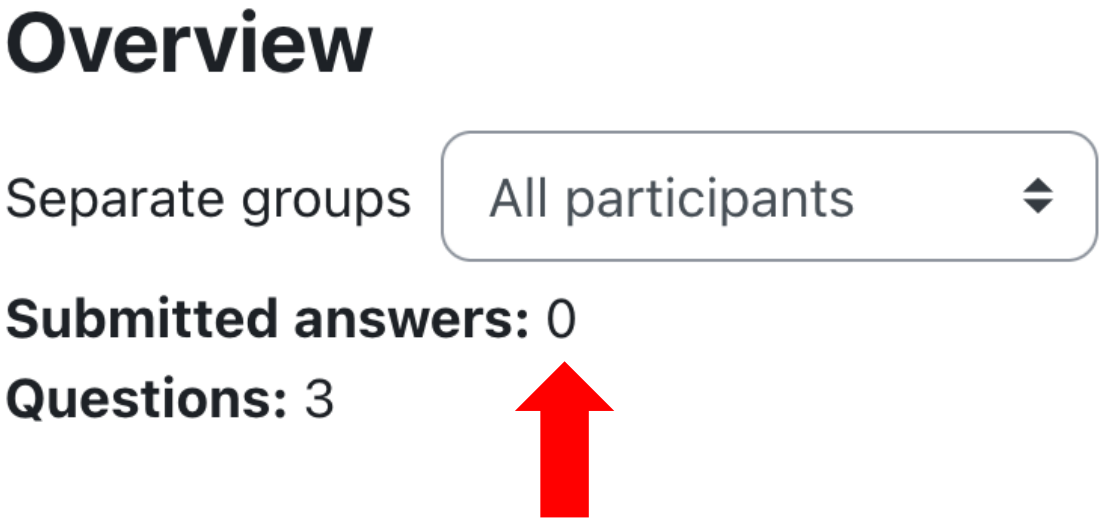
No course (recorded version available)

No case studies (the three classical case studies will be published, three additional will be available using the chatbot)

Exercise session at 17:00

Week after: enjoy holidays

Feedbacks


Overview
Separate groups
All participants
Submitted answers: 0
Questions: 3

Feedbacks

From **today until midnight Sunday 12 October,** you will have an opportunity to give feedback on courses!

One single question!

Please DO IT!!!!!!!!

Feedbacks

«Magistrale» was last weekend

127 masters in robotics