CSCI/ECEN 3302
Introduction to Robotics

Alessandro Roncone

aroncone@colorado.edu https://hiro-group.ronc.one

Administrivia

- Grades
  - Lab 2 is out
  - HW1 is out
  - Lab 3 will be out soon
  - Next in line is Lab 4
- Lab 5 and HW2 are going on

Future Outlook

- Many extra-credit opportunities
  - Mid-semester review up later this week and open for one week
    - Entirely ANONYMOUS
    - If 90% of the enrolled students fill it out, everyone in the class will will be awarded an extra 10 points to their final cumulative homework score!!!
- Maja Mataric AMA
- Robotics AMA
- After next week
  - We will start delving deeper in final project work!
  - You should start thinking about it – assignment coming up soon!

Maja Mataric AMA

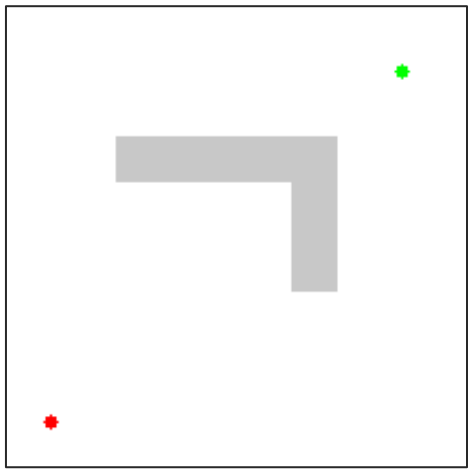
- Maja Mataric AMA
  - Video recording
  - Piazza discussion – extra credit for participating
  - Live AMA over Zoom

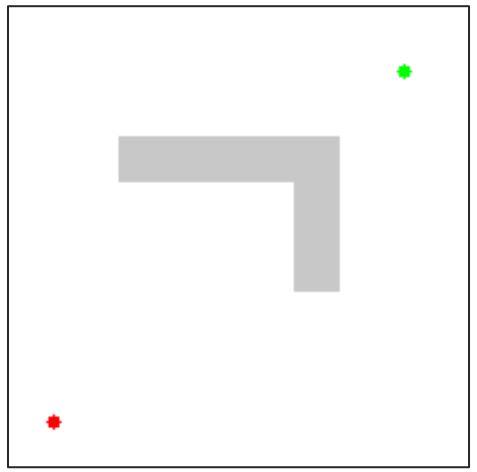
Robotics AMA

- Not mandatory
- Not recorded
- Loosely structured – I may have some slides or not
- I (we) will happily answer any questions you might have about robotics, in particular how to chart your path to a robotics career

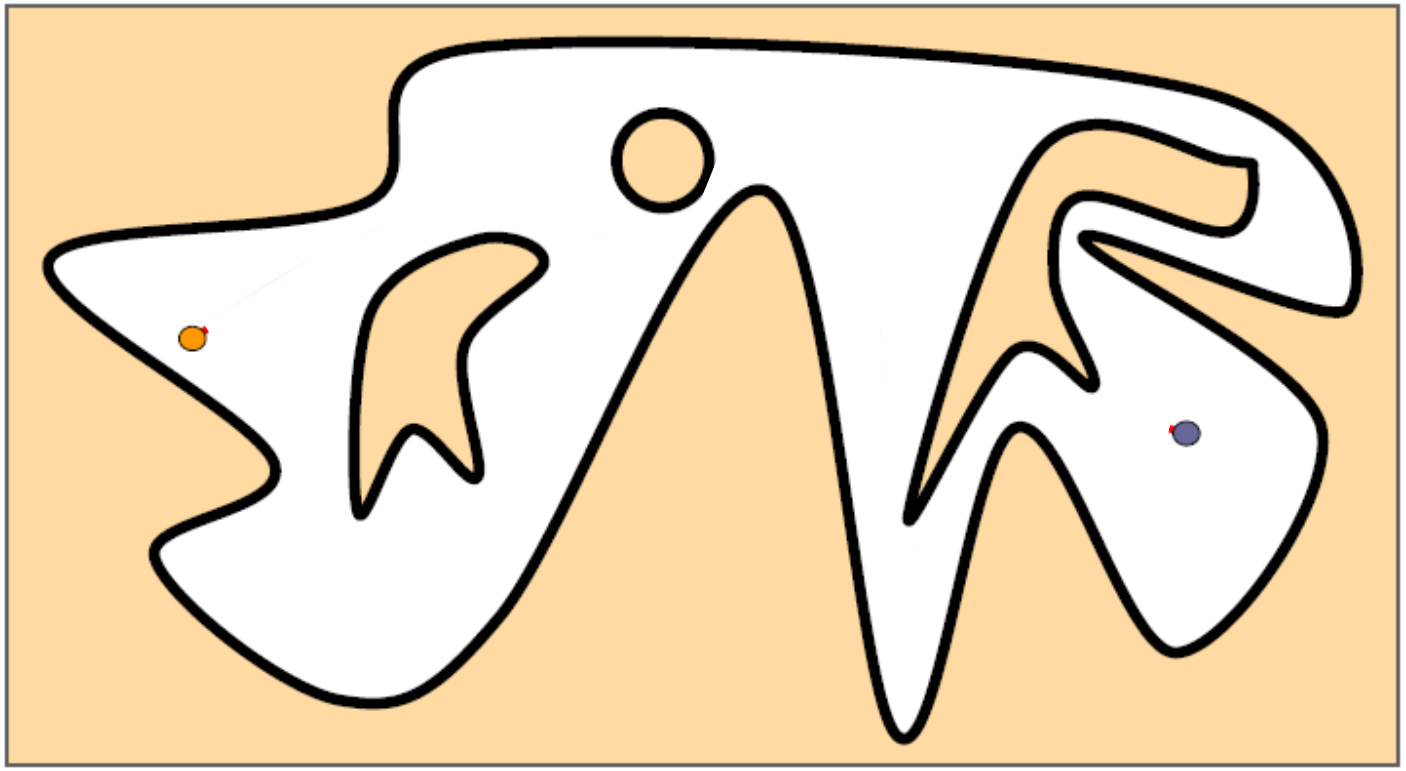
Chapter 13

RECAP
Search Algorithms





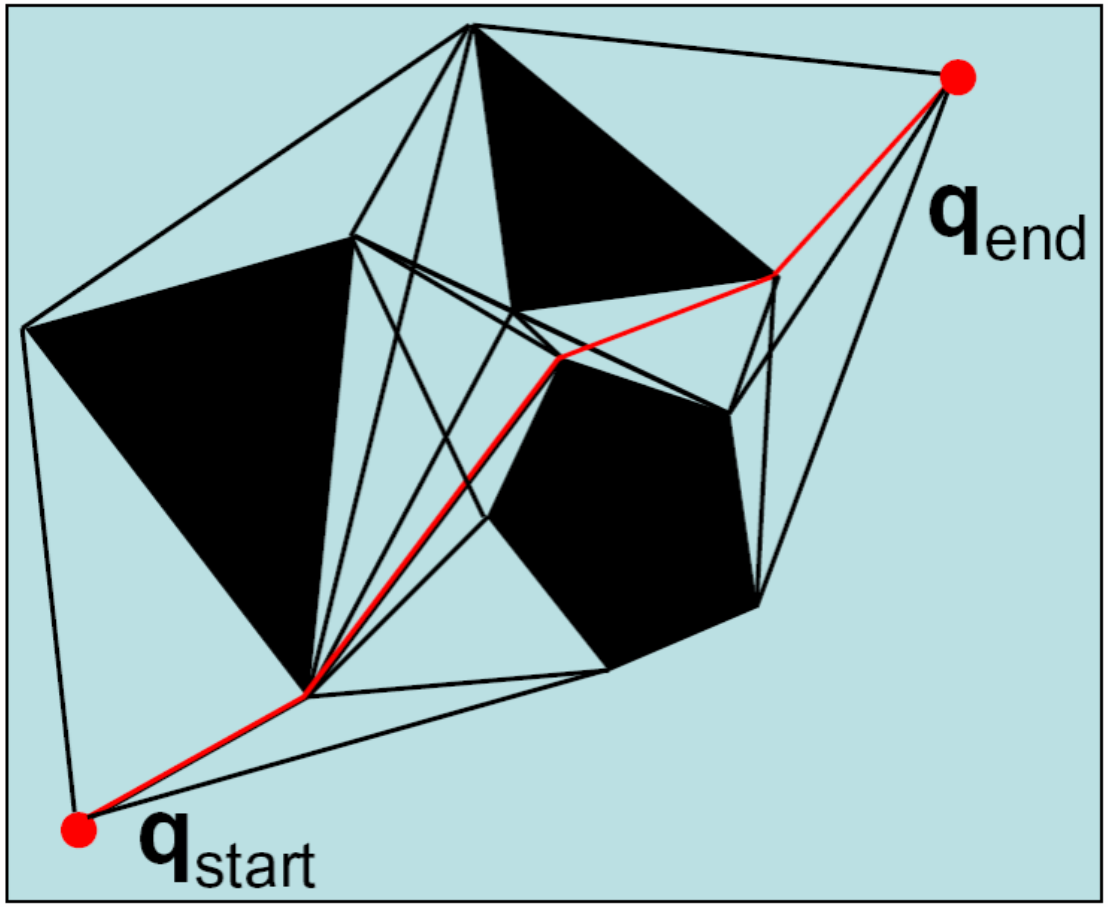
Moving from Start to Goal state



- General approach: Reduce what amounts to an intractable problem in continuous C-space to a tractable problem in a discrete space.
- Tools:
  - Environment Representations
  - Search Algorithms

Some Well-Known Representations

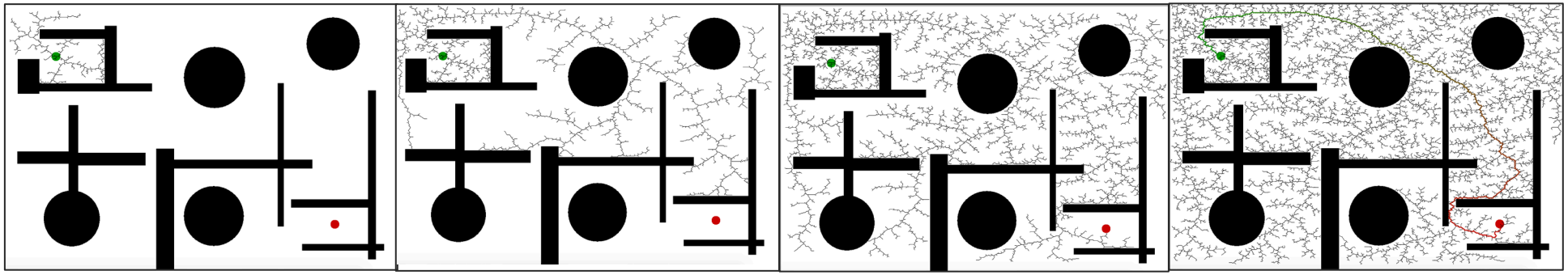
- Visibility Graphs
- Roadmap
- Cell Decomposition
- Potential Field



How to sample points?

- Uniformly vs randomly
- Sample more near places with few neighbors
- Bias samples to exist near obstacles
- Use human-provided waypoints
- Something better?

Rapidly-exploring Random Trees (RRT)



Rapidly Exploring Random Trees [RRT]

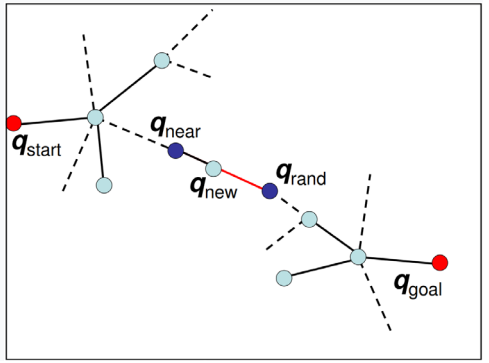
Basic idea:

- Build up a tree through generating “next states” in the tree by executing random controls
- However: to ensure good coverage, we need something better than the above!!

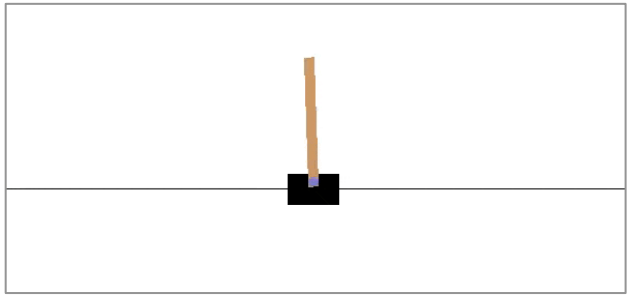
This is HW2!!

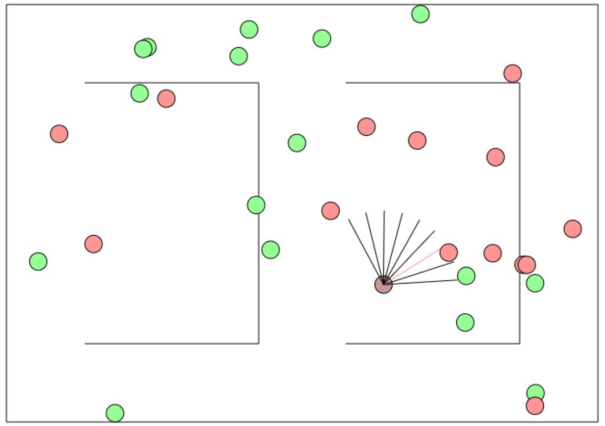
Rapidly-exploring Random Trees

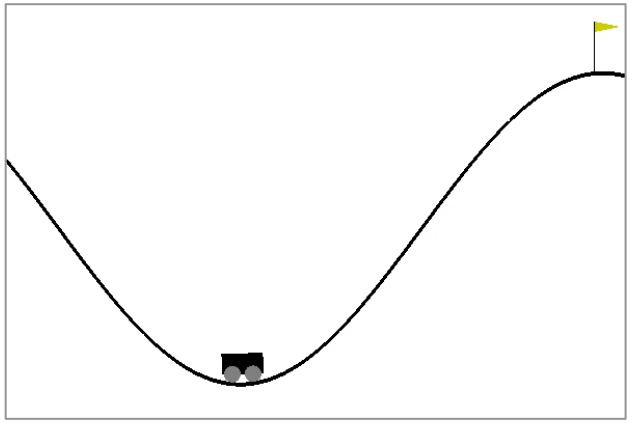
```
Algorithm BuildRRT
  Input: Initial configuration q_init, number of vertices in RRT K, incremental distance Δq)
  Output: RRT graph G

  G.init(q_init)
  for k = 1 to K
    q_rand ← RAND_CONF()
    q_near ← NEAREST_VERTEX(q_rand, G)
    q_new ← NEW_CONF(q_near, q_rand, Δq)
    G.add_vertex(q_new)
    G.add_edge(q_near, q_new)
  return G
```

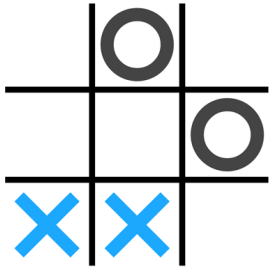


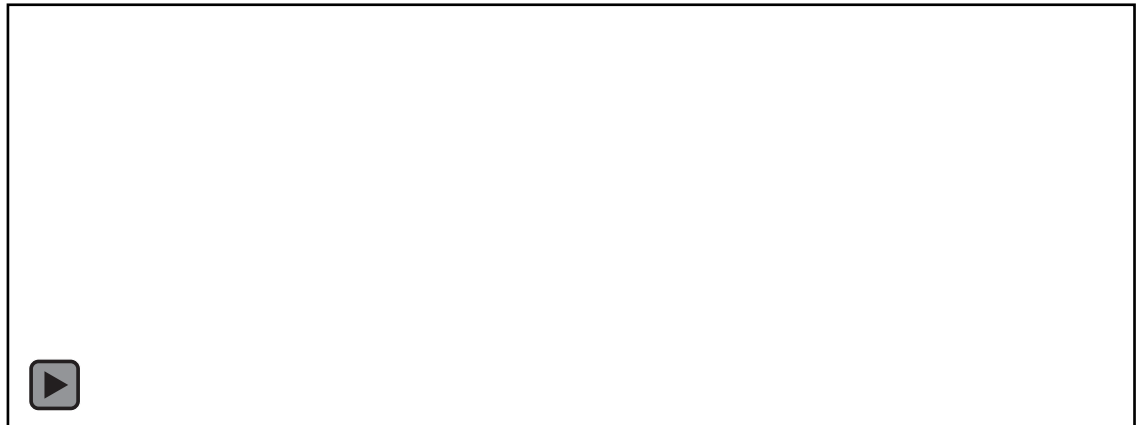
State Representation is Critical







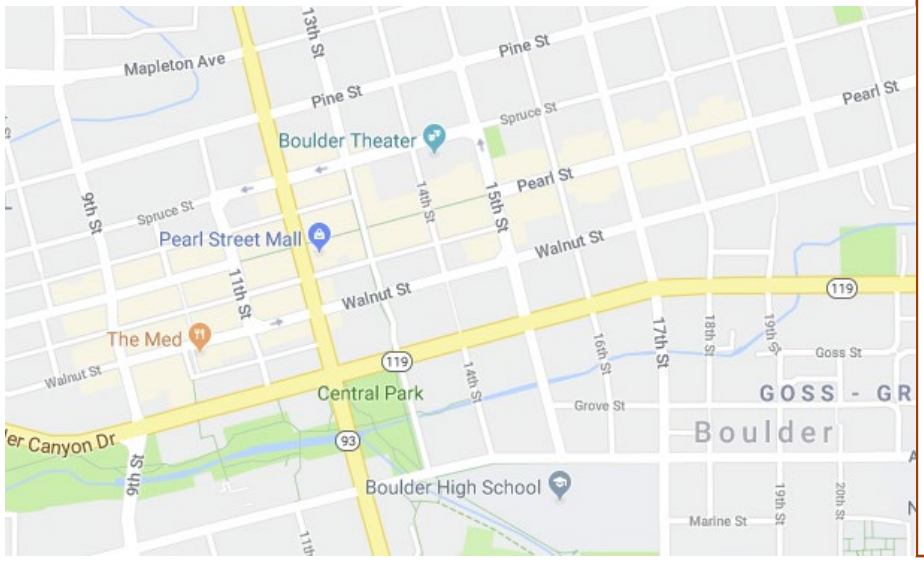


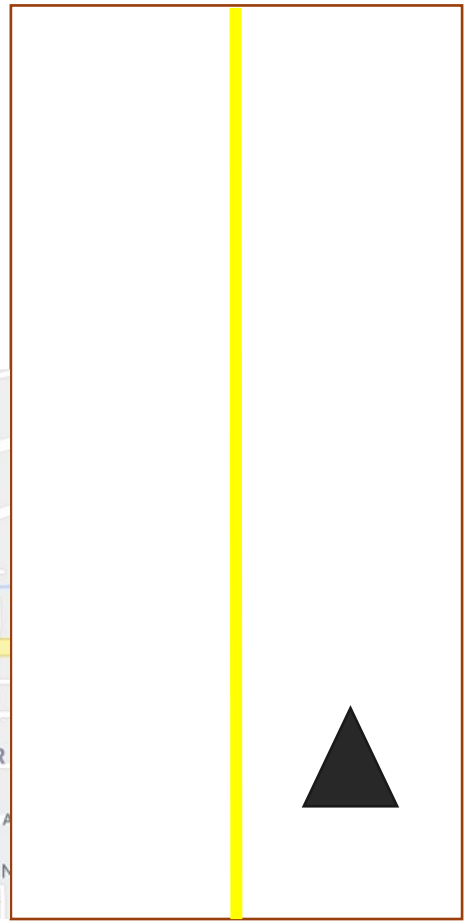


Exercise: Building an Autonomous Car

What do we need?

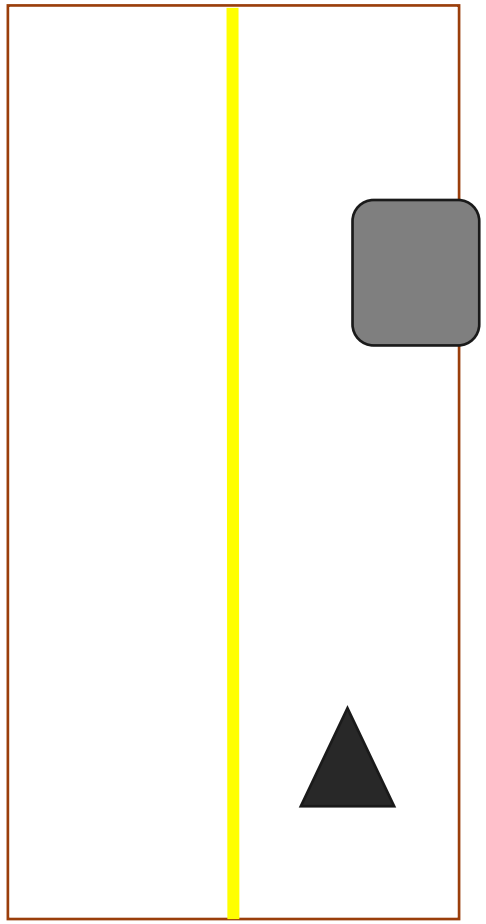
- World Representation
- Planning System





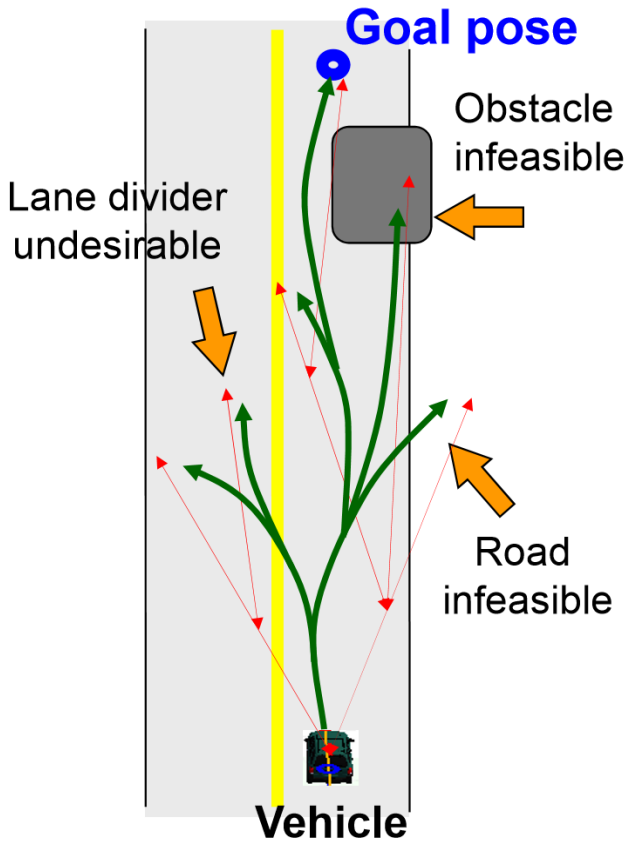
Exercise: Building an Autonomous Car

How can we get our agent to accommodate obstacles?



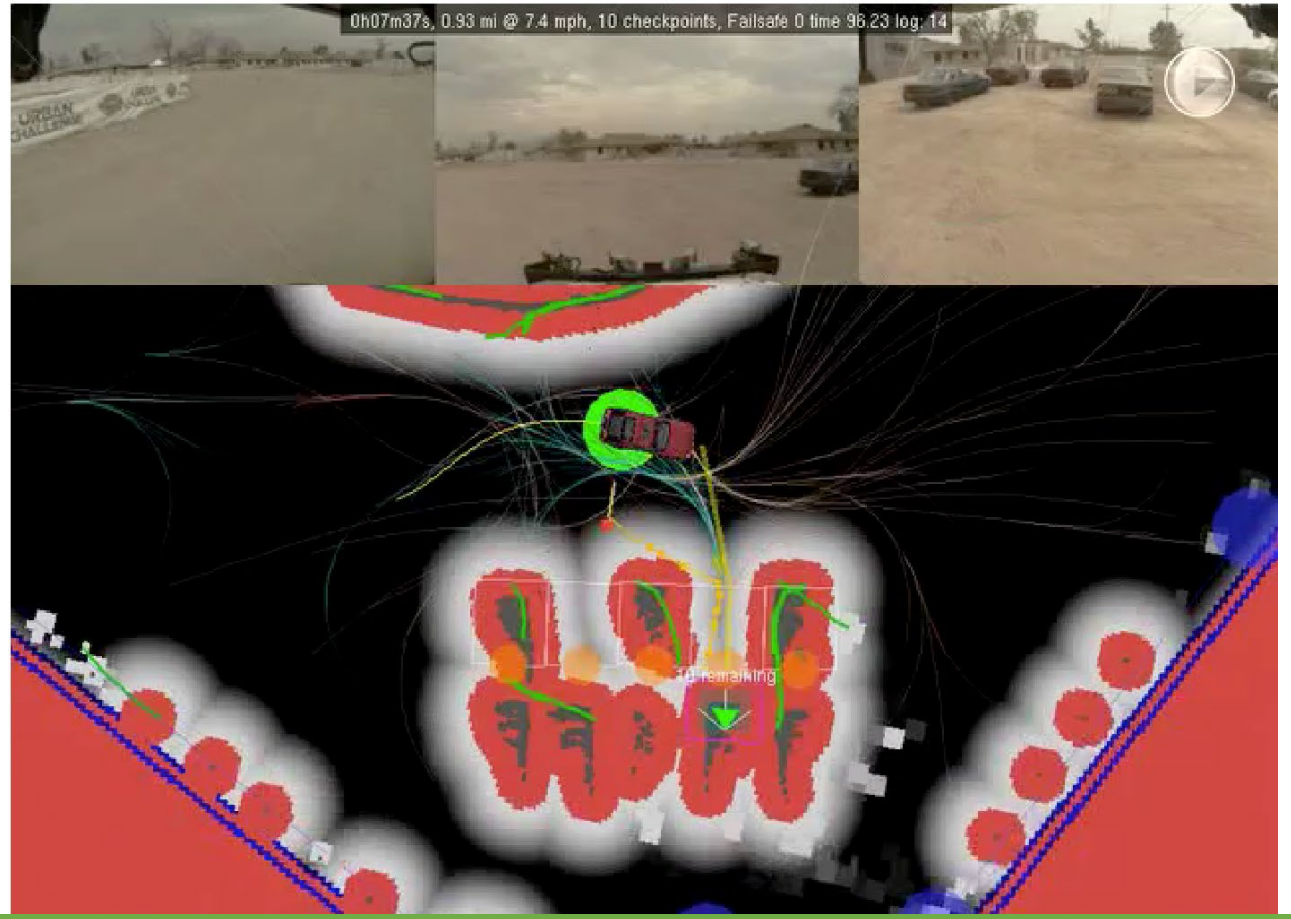
Exercise: Building an Autonomous Car

- Tree of trajectories is grown by sampling configurations randomly
- Rapidly explores several configurations that the robot can reach
  - Many test trajectories generated (10k/s)
- Safety of any trajectory is "guaranteed" as of instantaneous world state at the time of trajectory generation
- Choose best one that reaches the goal, e.g.:
  - Maximizes minimum distance to obstacles
  - Minimizes path length
- Supports dynamic replanning; if traj becomes infeasible:
  - Choose another that is feasible
  - If none remain, then e-stop



RRT for Autonomous Parking

http://acl.mit.edu/papers/KuwataIROS08.pdf



How can we make use of these representations?

Search algorithms provide a way to find a path!

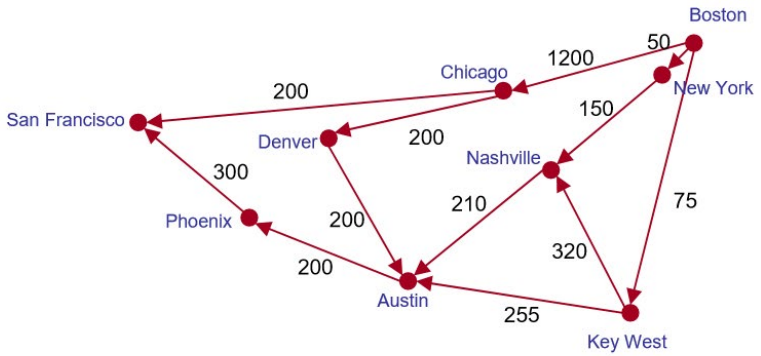
Search Algorithms

Allow us to use the world representations:

- Basics
- Uniform Cost
- Informed Search

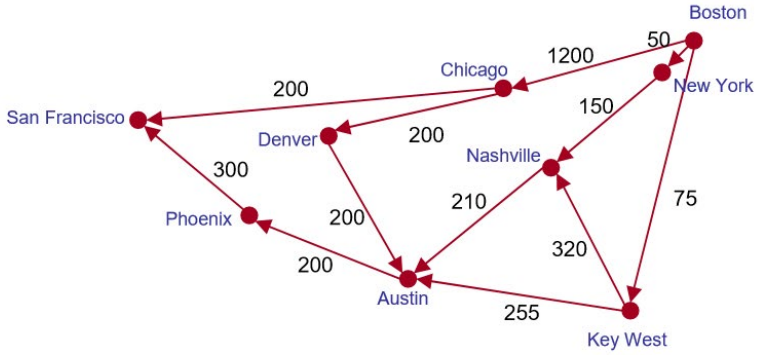
Goal Formulation

- Well defined function that identifies both the goal states and the conditions under which to achieve it
  - E.g. “Fly from Boston to San Francisco”
- Metrics for optimality may depend on:
  - Least amount of money
  - Fewest number of transfers
  - Shortest amount of time in the air
  - Shortest amount of time in airports

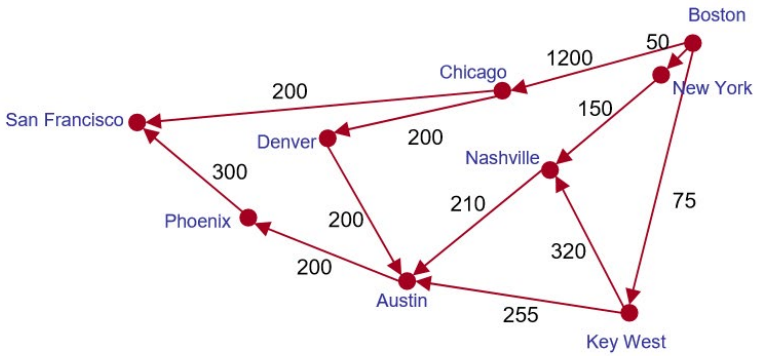


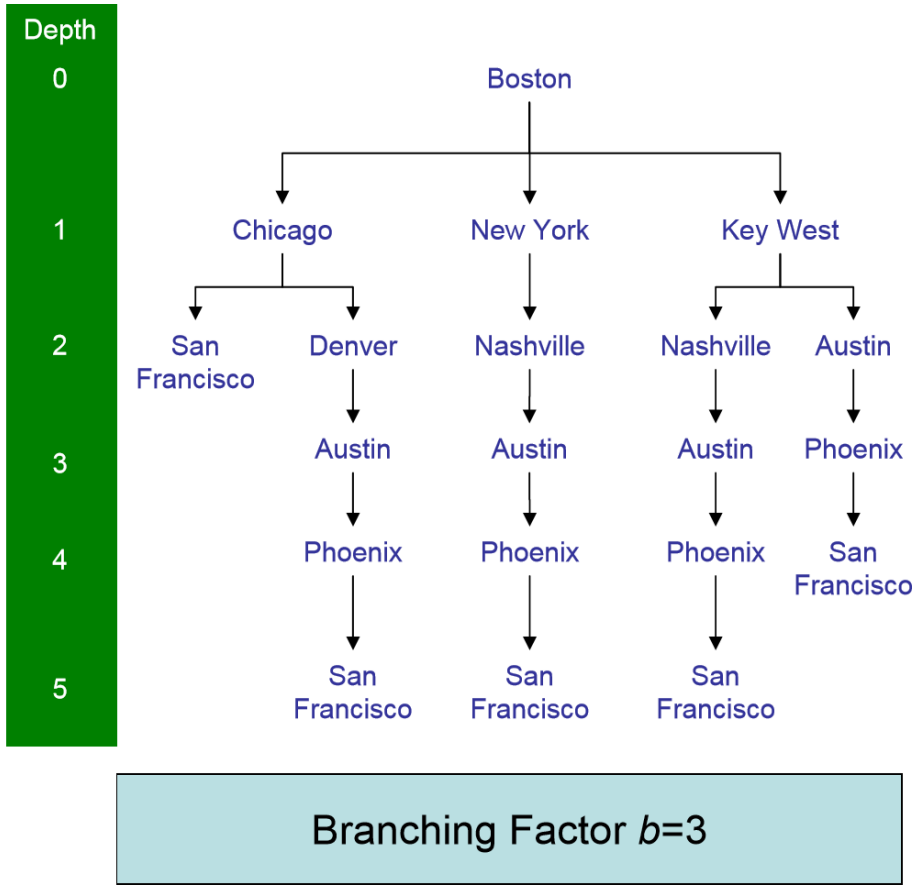
Problem formulation

- Well defined problems
  - Fully observable
  - Deterministic
  - Discrete Set of possible actions
- State Space: the set of all states that are reachable from an initial state by any sequence of actions
- Path: sequence of actions leading from one state to another

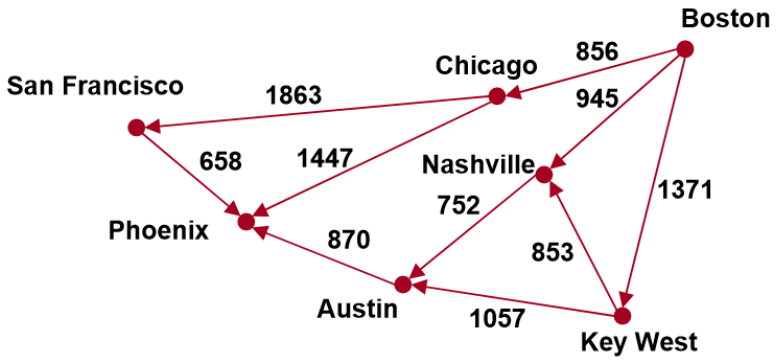


Search as a problem solving technique!!



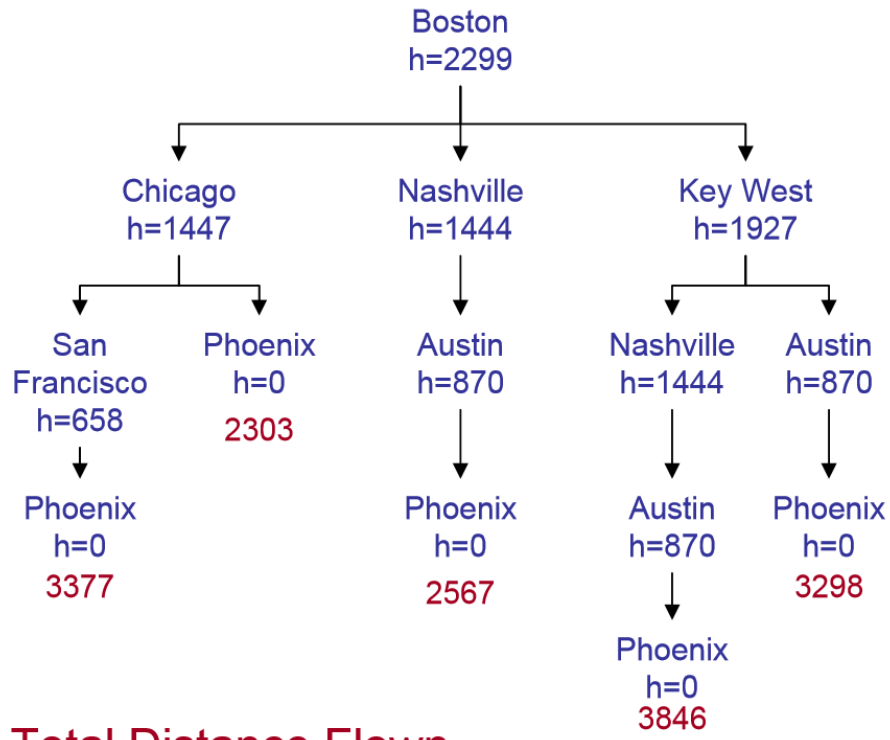


Greedy Best-First-Search



	Straight Line Distance to Phoenix
Boston	2299
Chicago	1447
Nashville	1444
Key West	1927
Austin	870
San Francisco	658

- Minimize estimated cost to reach a goal (in this case, the distance to Phoenix)



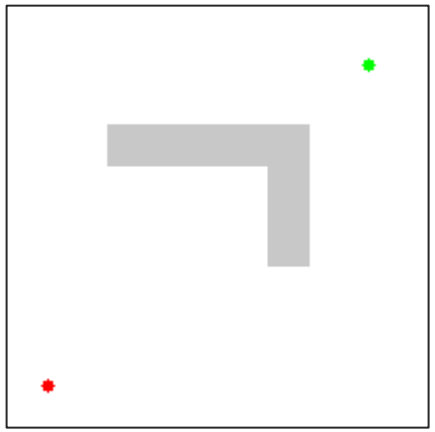
Total Distance Flown

Search Algorithms: Uniform Cost

Dijkstra's Algorithm:

```
1  function Dijkstra(Graph, source):
2
3      create vertex set Q
4
5      for each vertex v in Graph:             // Initialization
6          dist[v] ← INFINITY                  // Unknown distance from source to v
7          prev[v] ← UNDEFINED                 // Previous node in optimal path from source
8          add v to Q                          // All nodes initially in Q (unvisited nodes)
9
10     dist[source] ← 0                        // Distance from source to source
11
12     while Q is not empty:
13         u ← vertex in Q with min dist[u]    // Node with the least distance
14                                             // will be selected first
15         remove u from Q
16
17         for each neighbor v of u:           // where v is still in Q.
18             alt ← dist[u] + length(u, v)
19             if alt < dist[v]:               // A shorter path to v has been found
20                 dist[v] ← alt
21                  prev[v] ← u
22
23     return dist[], prev[]
```

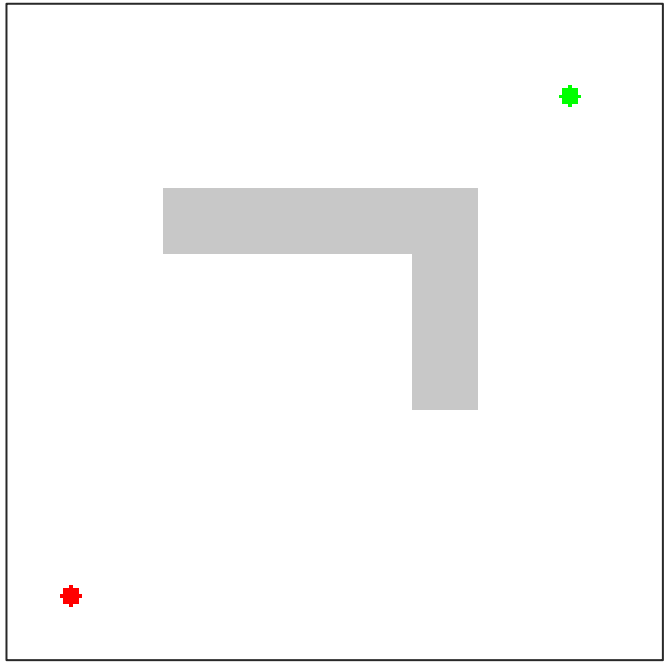
Finds shortest paths between nodes in a graph



```
1  S ← empty sequence
2  u ← target
3  if prev[u] is defined or u = source:          // Do something only if the vertex is reachable
4      while u is defined:                       // Construct the shortest path with a stack S
5          insert u at the beginning of S        // Push the vertex onto the stack
6          u ← prev[u]                           // Traverse from target to source
```

Informed Search

Finds shortest path between two nodes in a graph

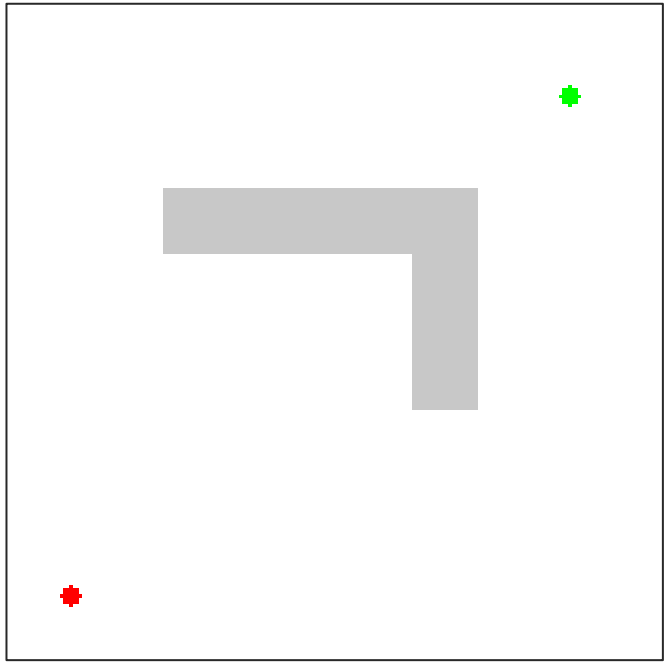


- Dijkstra's Algorithm:
- Lots of wasted exploration!
- We usually know which two nodes we want to find a path between:
  - Why compute all distances from source?

- Heuristic: A function that ranks choices in a search algorithm to help choose how to branch/explore.

A* Search Algorithm

Finds shortest path between two nodes in a graph



- Adds heuristic to Dijkstra's Algorithm

  to bias exploration!

- Insight:

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

- $f(n)$ = Cost so far + Est. cost to go

Admissible Heuristic: An **underestimate** of the shortest possible path from node n to the goal.

A* Search Algorithm

- Complete: Yes

  (If an answer exists, this will find it)

- Optimal: Yes

  (Returned answer will be the best possible)

- Optimally Efficient: Yes

  (No algorithm with same heuristic will expand fewer nodes)

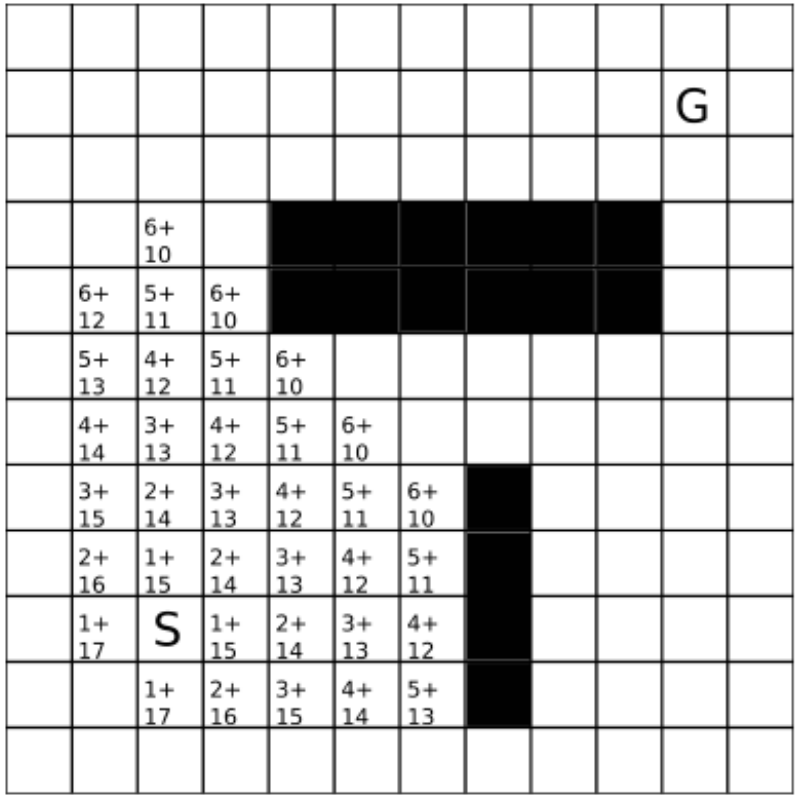
- Finding good admissible heuristics is challenging!

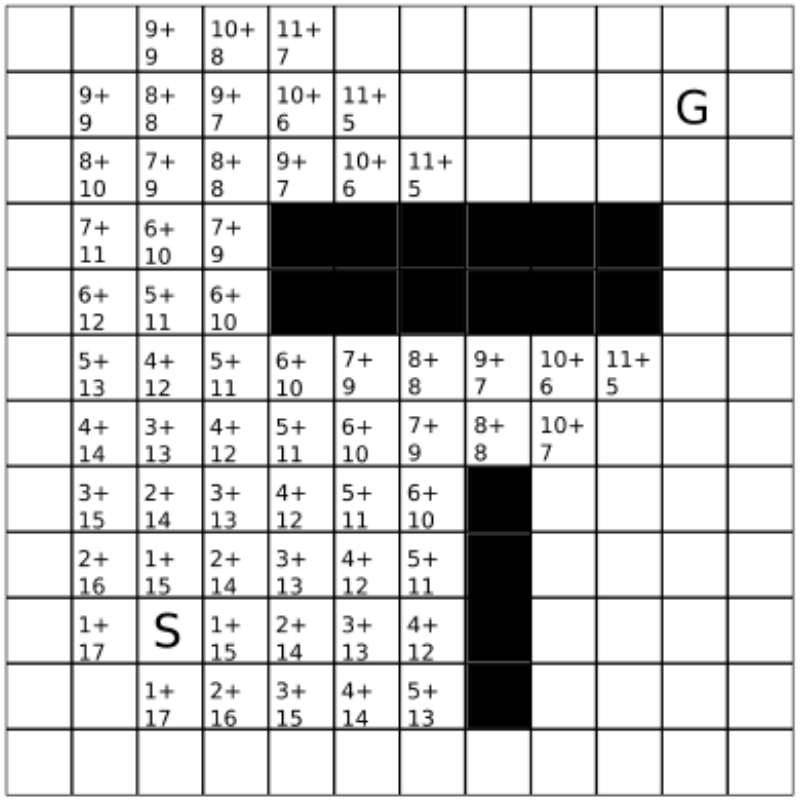
```
function reconstruct_path(cameFrom, current)
    total_path := [current]
    while current in cameFrom.Keys:
        current := cameFrom[current]
        total_path.append(current)
    return total_path
```

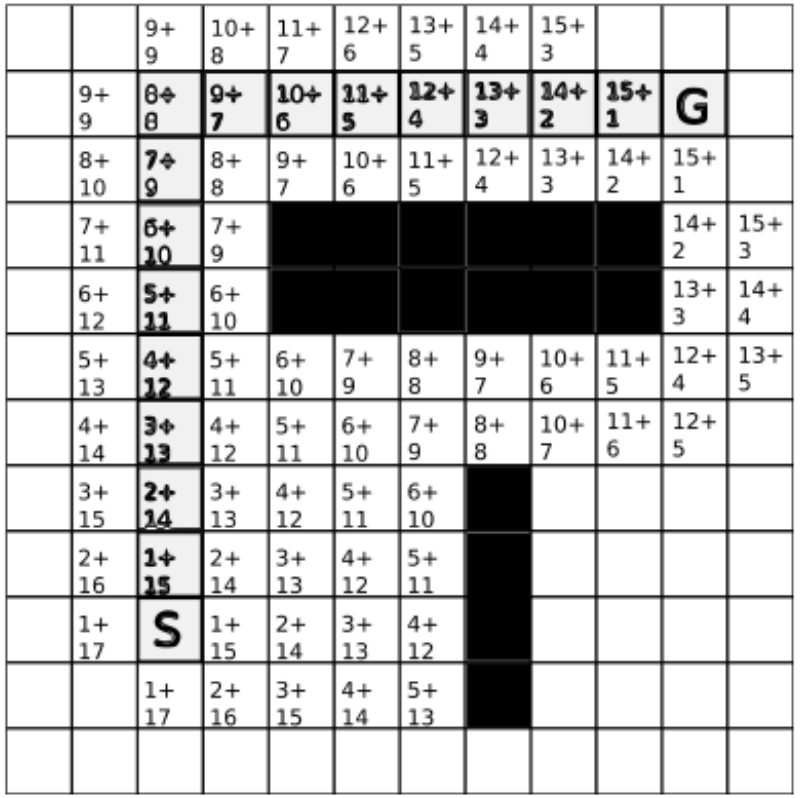
```
function A*(start, goal)
    // The set of nodes already evaluated
    closedSet := {}

    // The set of currently discovered nodes that are not evaluated yet.
    // Initially, only the start node is known.
    openSet := {start}

    // For each node, which node it can most efficiently be reached from.
    // If a node can be reached from many nodes, cameFrom will eventually contain the
    // most efficient previous step.
    cameFrom := the empty map

    // For each node, the cost of getting from the start node to that node.
    gScore := map with default value of Infinity

    // The cost of going from start to start is zero.
    gScore[start] := 0

    // For each node, the total cost of getting from the start node to the goal
    // by passing by that node. That value is partly known, partly heuristic.
    fScore := map with default value of Infinity

    // For the first node, that value is completely heuristic.
    fScore[start] := heuristic_cost_estimate(start, goal)

    while openSet is not empty
        current := the node in openSet having the lowest fScore[] value
        if current = goal
            return reconstruct_path(cameFrom, current)

        openSet.Remove(current)
        closedSet.Add(current)

        for each neighbor of current
            if neighbor in closedSet
                continue        // Ignore the neighbor which is already evaluated.

            if neighbor not in openSet  // Discover a new node
                openSet.Add(neighbor)

            // The distance from start to a neighbor
            tentative_gScore := gScore[current] + dist_between(current, neighbor)
            if tentative_gScore >= gScore[neighbor]
                continue        // This is not a better path.

            // This path is the best until now. Record it!
            cameFrom[neighbor] := current
            gScore[neighbor] := tentative_gScore
            fScore[neighbor] := gScore[neighbor] + heuristic_cost_estimate(neighbor, goal)

    return failure
```

Questions?

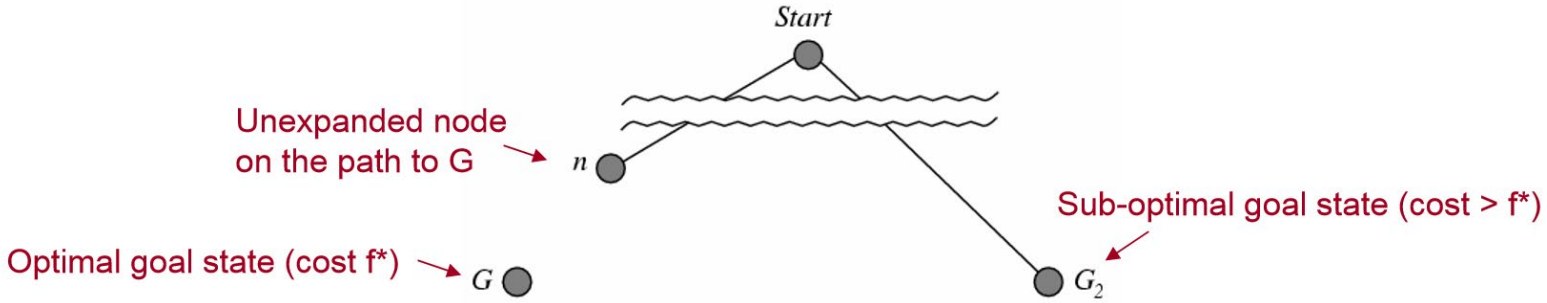
Dijkstra plus directional heuristic: A*







Optimality of A*



- Assume that G_2 has been chosen for expansion over n
- Because h is admissible:

$$f^* \geq f(n)$$

- If n is not chosen for expansion over G_2, we must have:

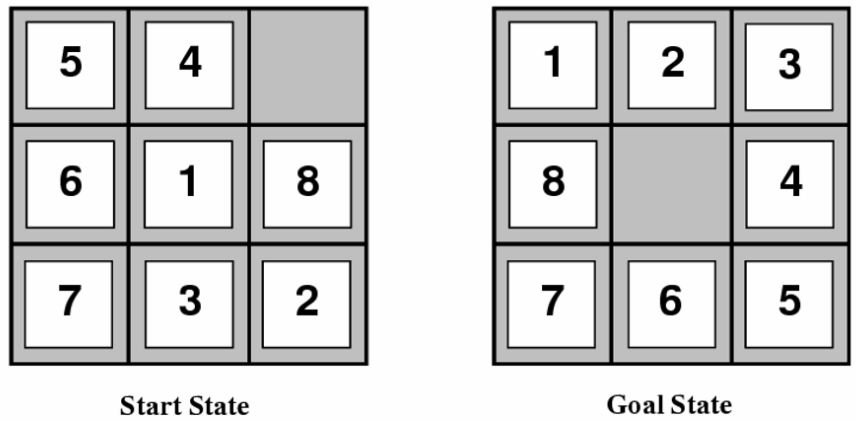
$$f(n) \geq f(G_2)$$

- Combining these, we get:

$$f* \geq f(G_2)$$

- However, this violates the assertion that G_2 is sub-optimal! Therefore, A^* never selects a sub-optimal goal for expansion

Heuristic Selection



- Must be admissible (never over-estimate)
- Heuristics for the 8-Puzzle

Heuristic Selection Effects

	Search Cost			Effective Branching Factor		
d	IDS	A*(h_1)	A*(h_2)	IDS	A*(h_1)	A*(h_2)
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	364404	227	73	2.78	1.42	1.24
14	3473941	539	113	2.83	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

- Compare iterative-deepening with A* using
  h1 (# misplaced tiles) and *h2* (city block distance)
- Effective branching factor b*
  - Number of expanded nodes = $1 + b^* + (b^*)^2 + \ldots + (b^*)^{depth}$
  - b* remains relatively constant across many measurements

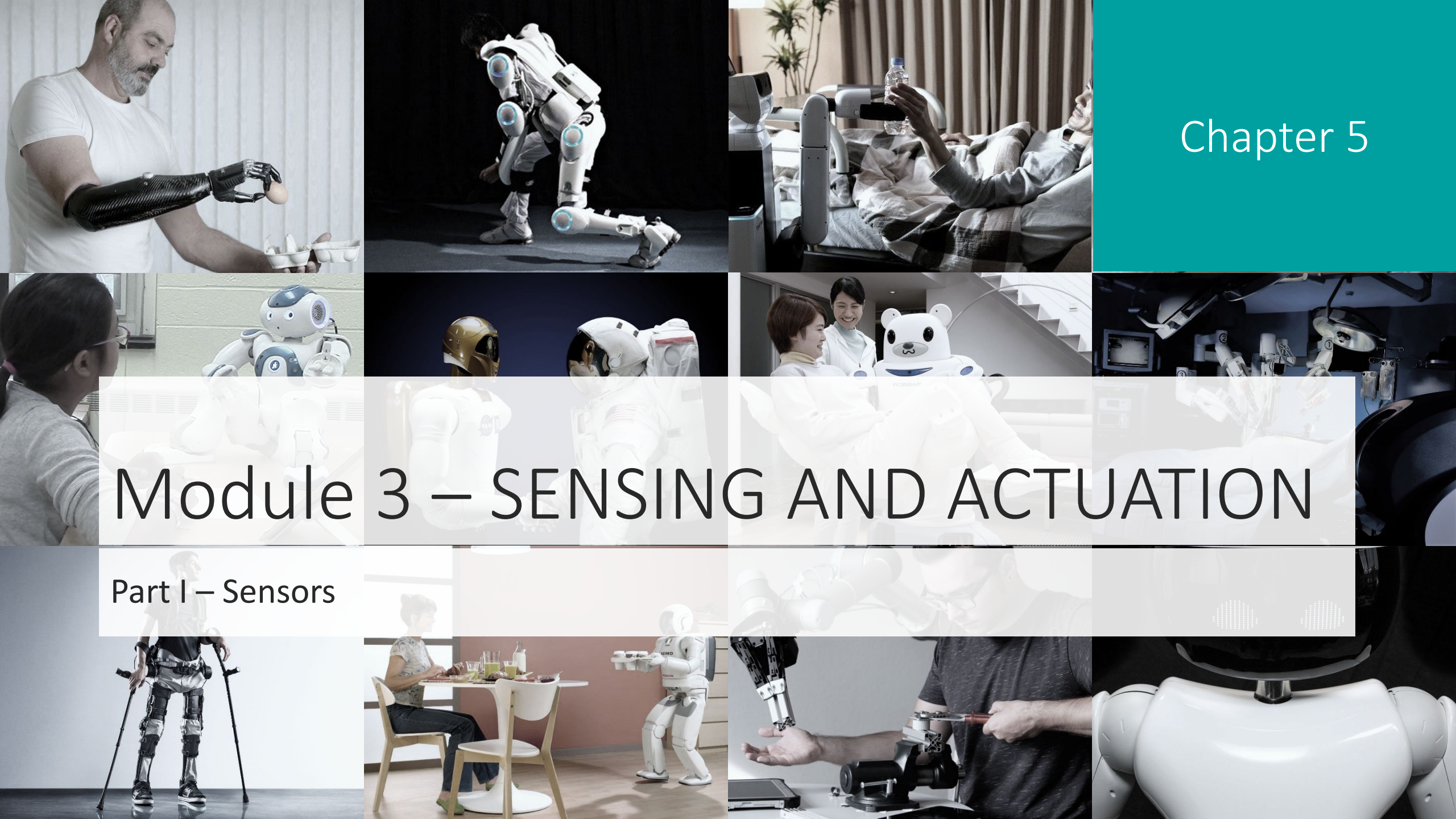
How do humans solve the path/motion planning problem?

Wrap-up

	Completeness	Computational Complexity
Exact Cell Decomposition	Complete	Exponential time
Potential Fields	Not complete	Polynomial time
Discretization and A* search	Resolution complete	Exponential time
Probabilistic RoadMap (PRM)	Probabilistically complete	Hard to characterize!

Take-home lessons

- First step in addressing a planning problem is choosing a suitable map representation
- Reduce robot to a point-mass by inflating obstacles
- Grid-based algorithms are complete, sampling-based ones probabilistically complete, but usually faster
- Most real planning problems require combination of multiple algorithms

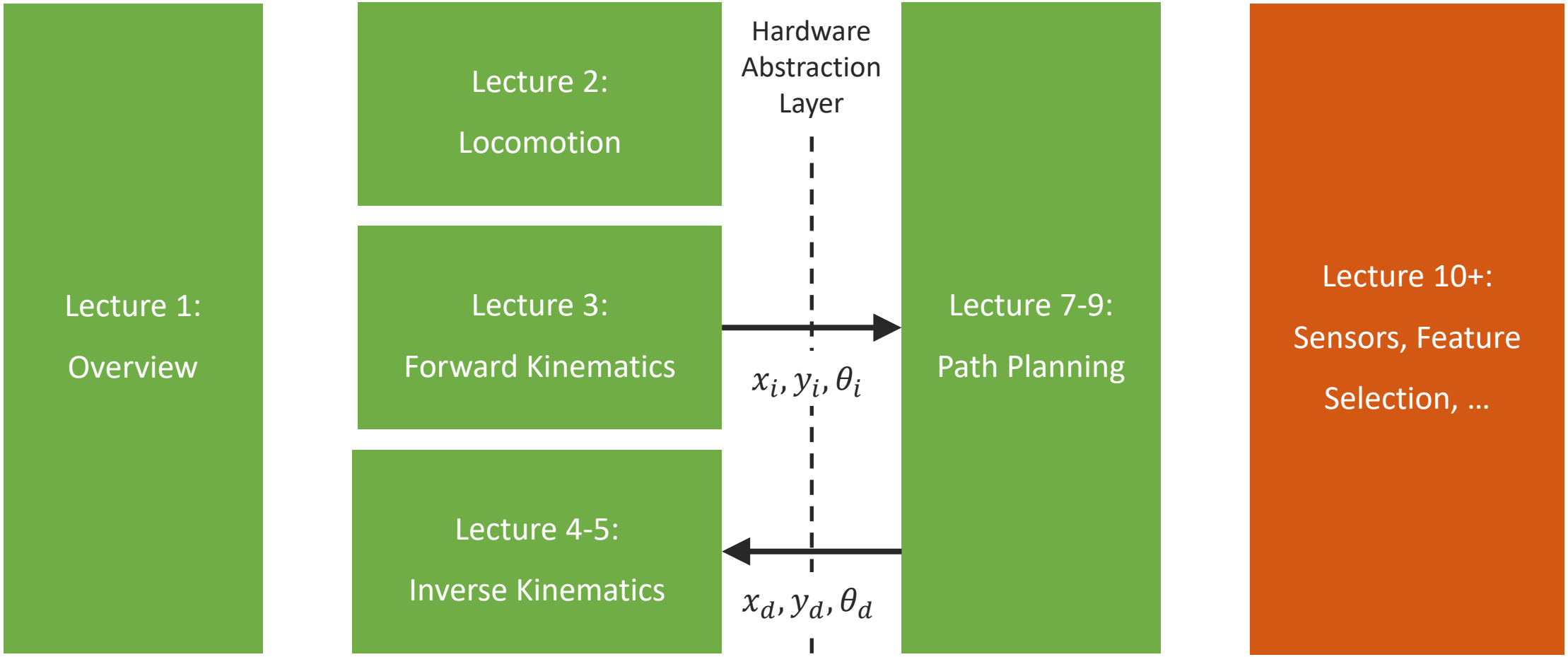


Chapter 5

Module 3 – SENSING AND ACTUATION

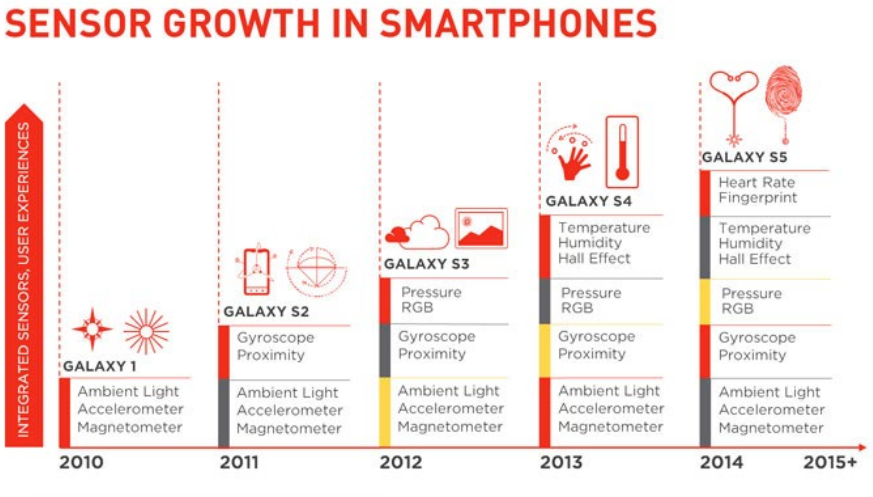
Part I – Sensors

Roadmap


Lecture 1:
Overview
Lecture 2:
Locomotion
Lecture 3:
Forward Kinematics
Lecture 4-5:
Inverse Kinematics
Hardware
Abstraction
Layer
x_i, y_i, θ_i
x_d, y_d, θ_d
Lecture 7-9:
Path Planning
Lecture 10+:
Sensors, Feature
Selection, …

Sample of everyday sensors

Camera



US (ultrasonic)

3D

PIR
(passive infrared)

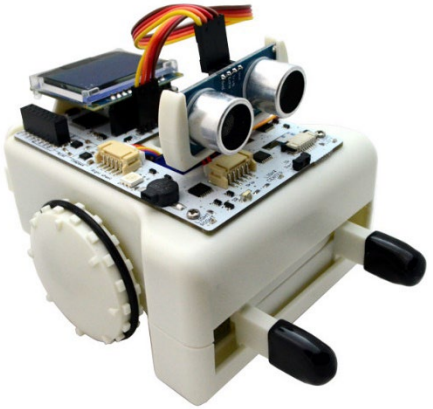
Radar

So far: Robotics always benefits, never drives sensor development

Sensors on ~~Sparki~~ e-puck

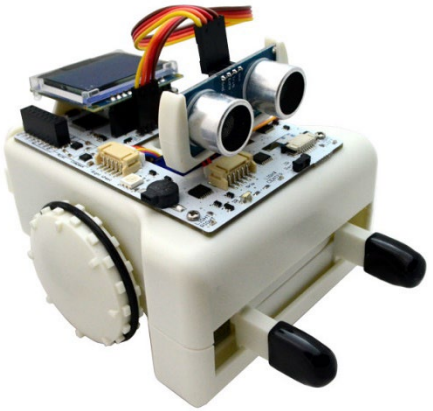
- Accelerometer
- Gyroscope
- Floor sensor
- Ultrasound sensor
- IR receiver
- Light sensor
- NO Encoders due to Stepper Motor

Which of these sensors can be used to reduce odometry error?



Improving Sparki's Odometry

- Accelerometer:
- Magnetometer:
- Floor sensor:
- Ultrasound sensor:
- IR receiver:
- …



Few (any?) sensors have specific applications, but most problems benefit from as much information as possible.

Classifying sensors

- Type of information
- Physical Principle
- Absolute vs. derivative
- Amount of information (Bandwidth)
- Low and high reading (Dynamic range)
- Accuracy and Precision

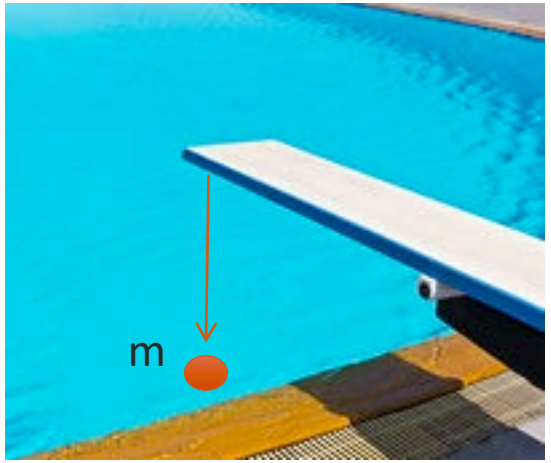
Specification	URG-04LX
Power source	Regulated 5V ±5%
Interface	RS232, USB
Detection Distance	20 to 4000 (mm)
Guaranteed Accuracy (min to 1m)	±10mm
Guaranteed Accuracy (1m to max)	1% of detected distance

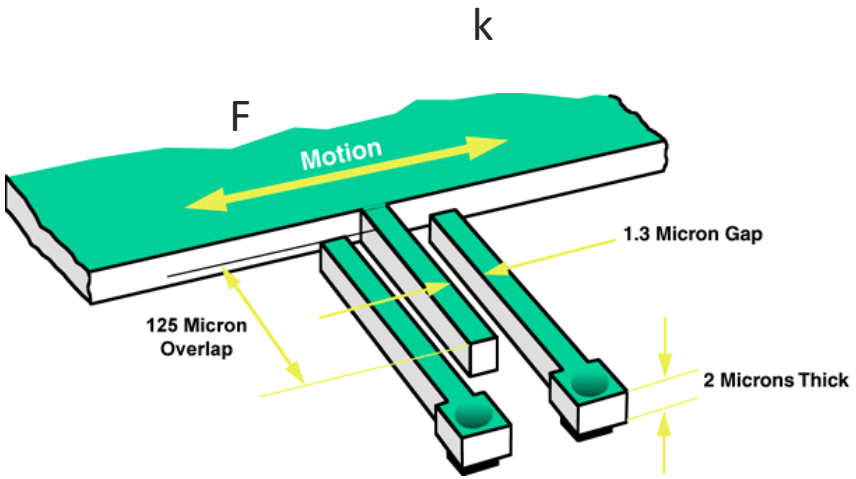
Specifications	
Power source	5V +/-5%
Current consumption	0.5A (Rush current 0.8A)
Detection range	0.02 to approximately 4m
Laser wavelength	785nm, Class 1
Scan angle	240°
Scan time	100msec/scan (10.0Hz)
Resolution	1mm
Angular Resolution	0.36°
Interface	USB 2.0, RS232
Weight	5.0 oz (141 gm)

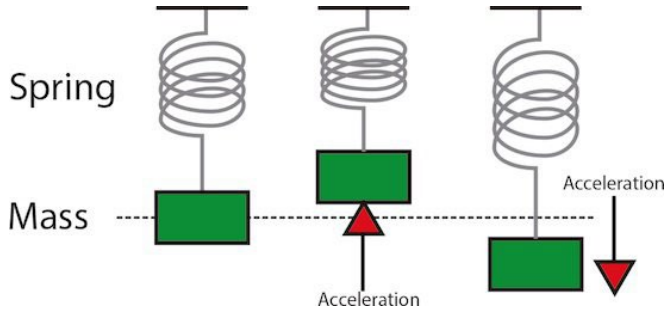
Accelerometer

$$F = kx = ma$$

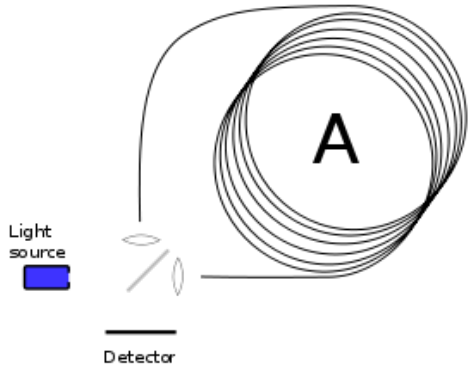
- Very cheap and small
- Measures acceleration
- Integrate for speed and distance
- Applications
  - Tell the pose of an object from the direction of gravity
  - Tell when robot hits an object



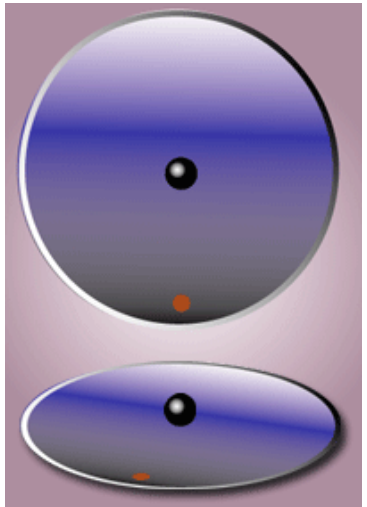




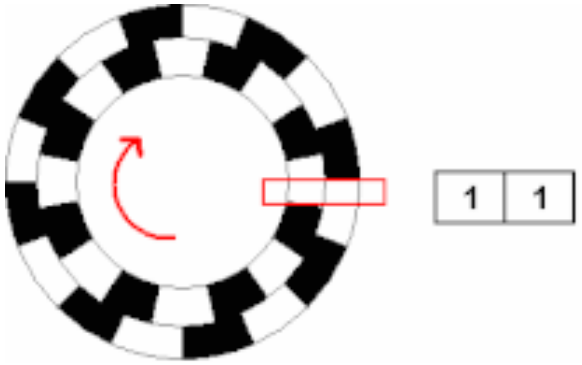
Gyroscopes

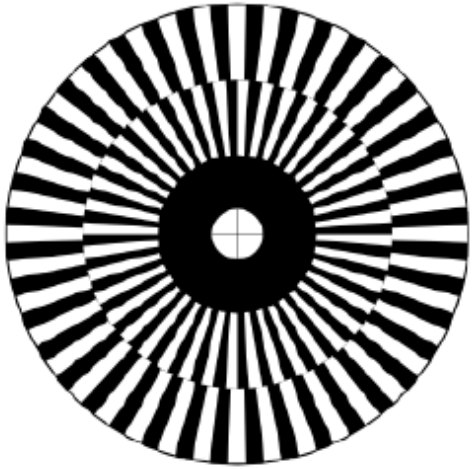


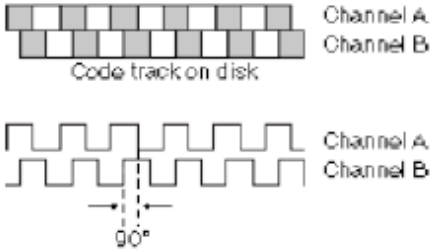
- Measures orientation
- Very expensive, infeasible to miniaturize
- Rate gyroscopes measure rotational speed
- Implemented using MEMS vibration devices, measure Coriolis force on two proof masses vibrating in plane
- Applications
  - Correct heading

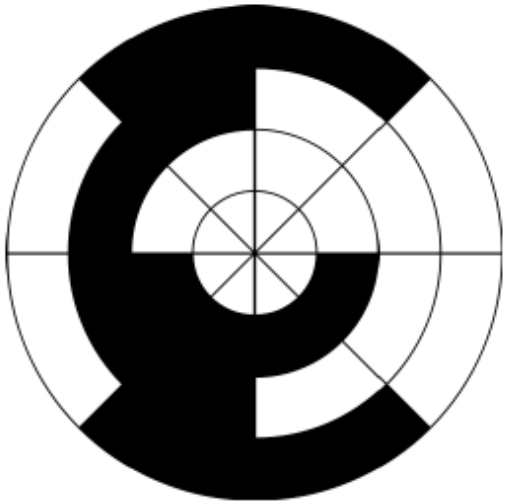


Wheel/Joint encoder



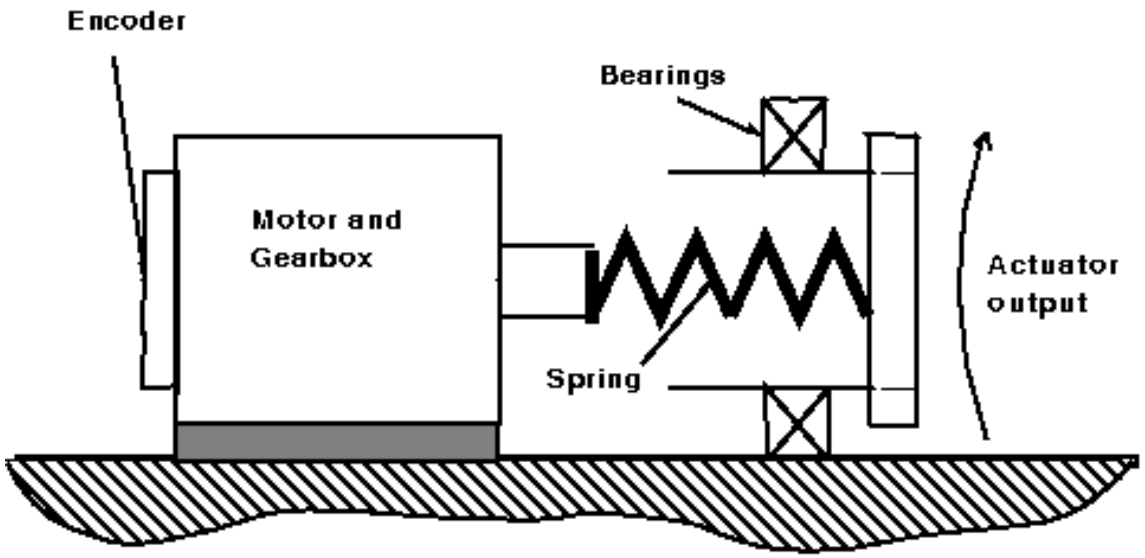


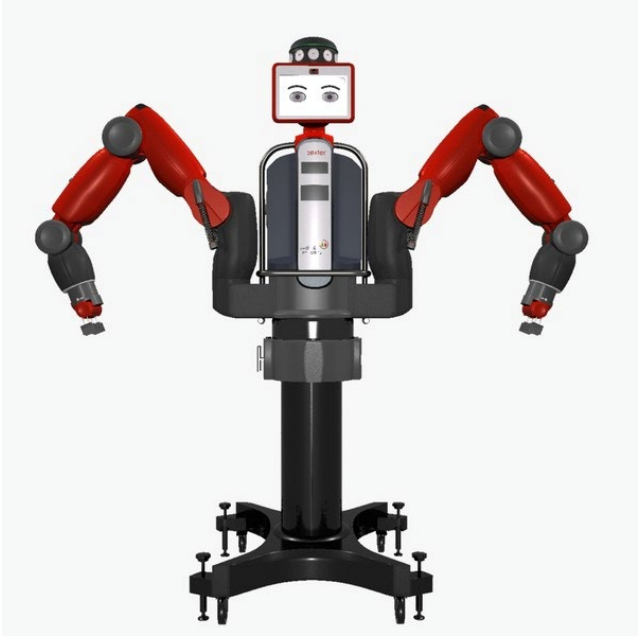




Can also be implemented magnetically or electrically (same principle). Main stream technology: CNC machines and RC servos.

Series Elastic Actuator





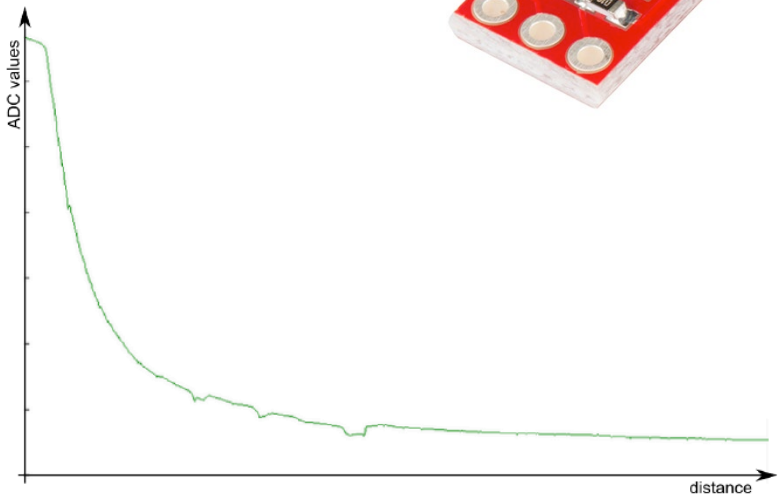
F=kx (Hooke's law)

Measure distance using potentiometer

Series Elastic Actuator

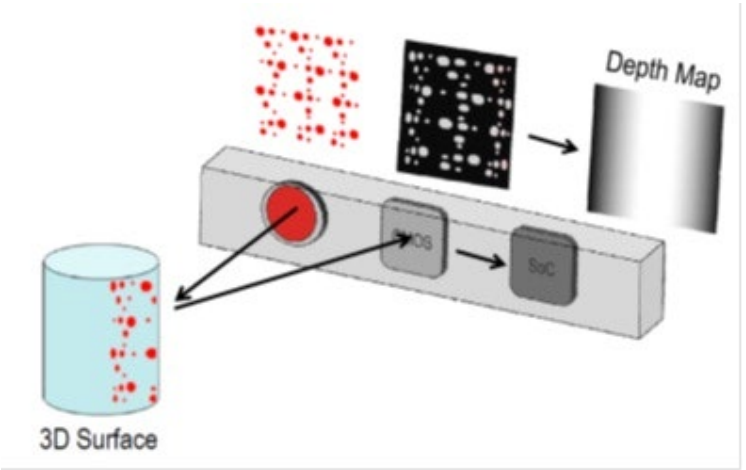
Distance from light intensity

- Emitter/receiver pair
- Highly non-linear
- Depends on surface color
- Confuses emissions from other sensors
- Requires a lot of energy



Distance from structure

Zhang et al. (2002)

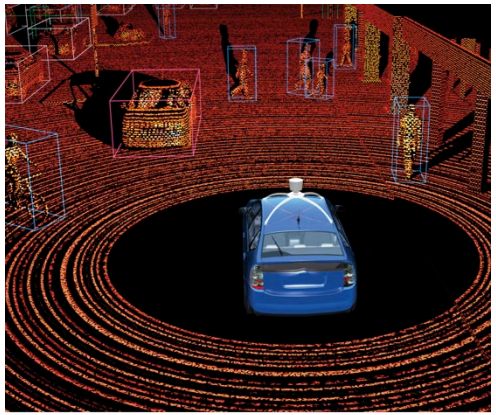


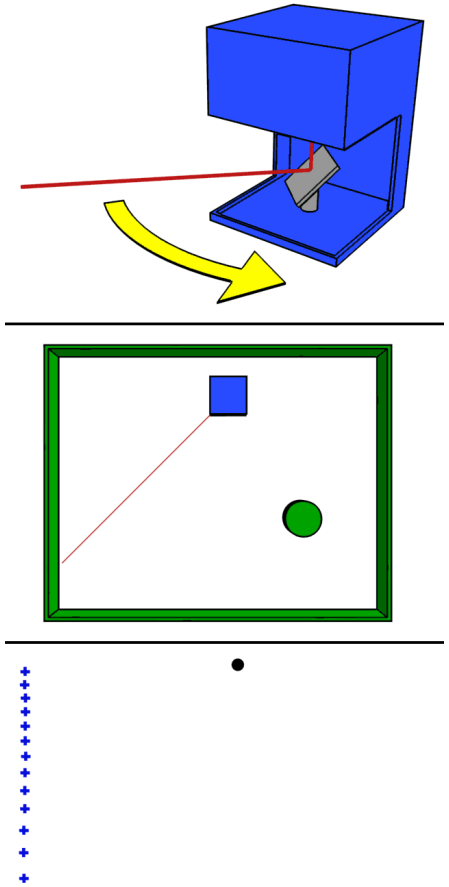
http://www.depthbiomechanics.co.uk/?p=100

Distance from Sound

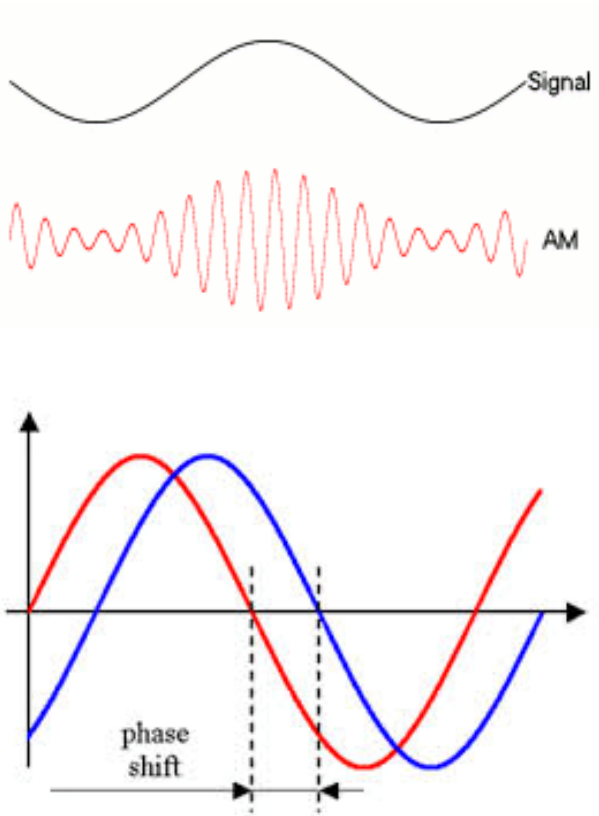
- Emitter/receiver pair
- Algorithm
  - Emit ping
  - Measure time until it returns
  - Calculate distance based on 300m/s
- Requires large objects
- Quality of result depends on object size

Distance from phase shift



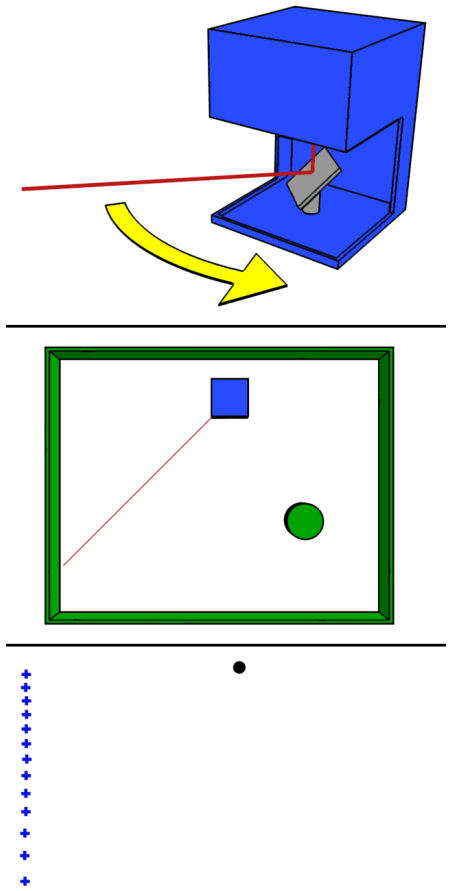


5 Mhz ~ 60m



https://en.wikipedia.org/wiki/Lidar (LIght Detection And Ranging)

How much data does a laser scanner produce?



Specifications	
Power source	5V +/-5%
Current consumption	0.5A (Rush current 0.8A)
Detection range	0.02 to approximately 4m
Laser wavelength	785nm, Class 1
Scan angle	240°
Scan time	100msec/scan (10.0Hz)
Resolution	1mm
Angular Resolution	0.36°
Interface	USB 2.0, RS232
Weight	5.0 oz (141 gm)

Specification	URG-04LX
Power source	Regulated 5V ±5%
Interface	RS232, USB
Detection Distance	20 to 4000 (mm)
Guaranteed Accuracy (min to 1m)	±10mm
Guaranteed Accuracy (1m to max)	1% of detected distance

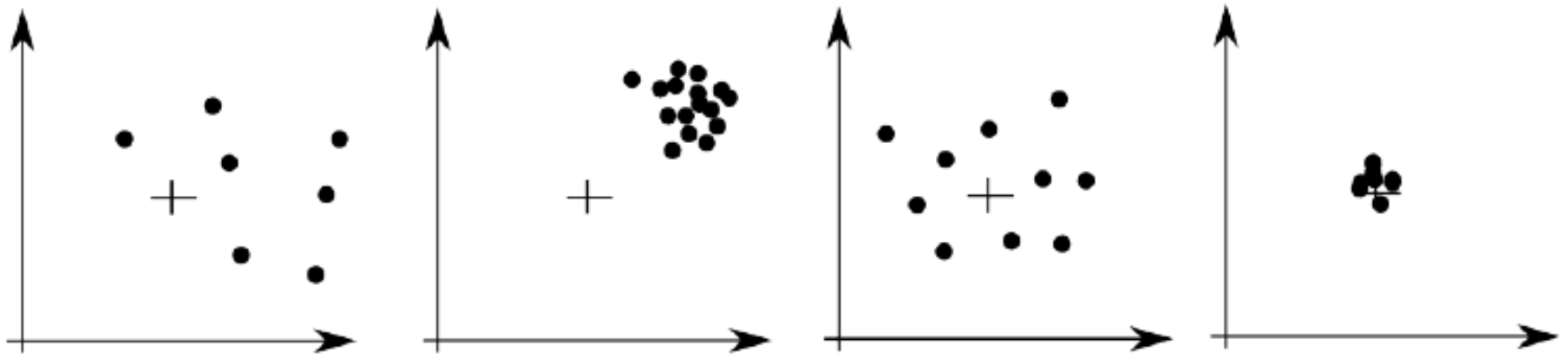
Hokuyo URG

What to do with so much data? (Cameras are even worse!)

Sensor characteristics

- Active and passive
- Resolution
- Accuracy
- Precision
- Bandwidth
- Range
- Dynamic Range
- Cross-sensitivity

Precision vs. Accuracy



Neither precise nor accurate, precise and not accurate, accurate and not precise, precise and accurate

Performance of a laser scanner

- Range: difference between highest and lowest reading
- Dynamic range: ratio of lowest and highest reading
- Resolution: minimum difference between values
- Linearity: variation of output as function of input
- Bandwidth: speed with which measurements are delivere
- Cross-Sensitivity: sensitivity to environment
- Accuracy: difference between measured and true value
- Precision: reproducibility of results

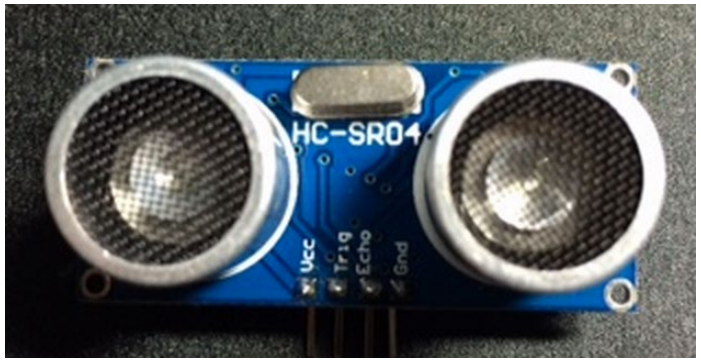
Hokuyo URG

Specification	URG-04LX
Power source	Regulated 5V ±5%
Interface	RS232, USB
Detection Distance	20 to 4000 (mm)
Guaranteed Accuracy (min to 1m)	±10mm
Guaranteed Accuracy (1m to max)	1% of detected distance

Specifications	
Power source	5V +/-5%
Current consumption	0.5A (Rush current 0.8A)
Detection range	0.02 to approximately 4m
Laser wavelength	785nm, Class 1
Scan angle	240°
Scan time	100msec/scan (10.0Hz)
Resolution	1mm
Angular Resolution	0.36°
Interface	USB 2.0, RS232
Weight	5.0 oz (141 gm)

Performance of an ultrasonic sensor

- Range: difference between highest and lowest reading
- Dynamic range: ratio of lowest and highest reading
- Resolution: minimum difference between values
- Linearity: variation of output as function of input
- Bandwidth: speed with which measurements are deliver
- Cross-Sensitivity: sensitivity to environment
- Accuracy: difference between measured and true value
- Precision: reproducibility of results



- Power Supply :+5V DC
- Quiescent Current : <2mA
- Working Currnt: 15mA
- Effectual Angle: <15°
- Ranging Distance : 2cm – 400 cm/1" - 13ft
- Resolution : 0.3 cm
- Measuring Angle: 30 degree
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm

Summary

- Sensors do not serve specific applications and no sensor solves a problem completely
- Many sensors observe the same phenomenon using different physical principles
- Different sensors have different trade-offs qualified in their different precision, accuracy, bandwidth, dynamic range and resolution
- There are smart ways to extract the desired information from a set of sensors and fuse them

  [Sensor Fusion]