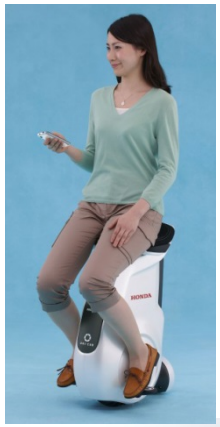
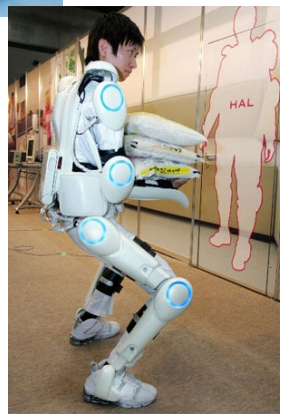
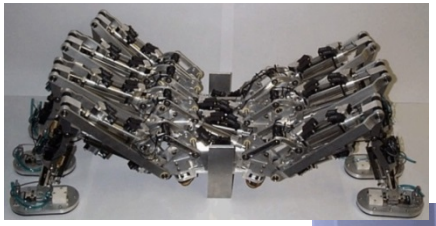
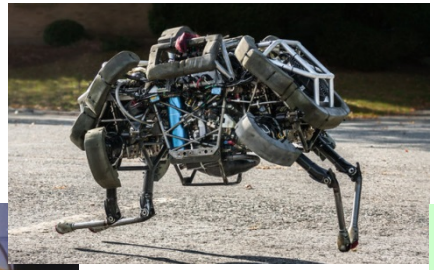
Localisation and navigation







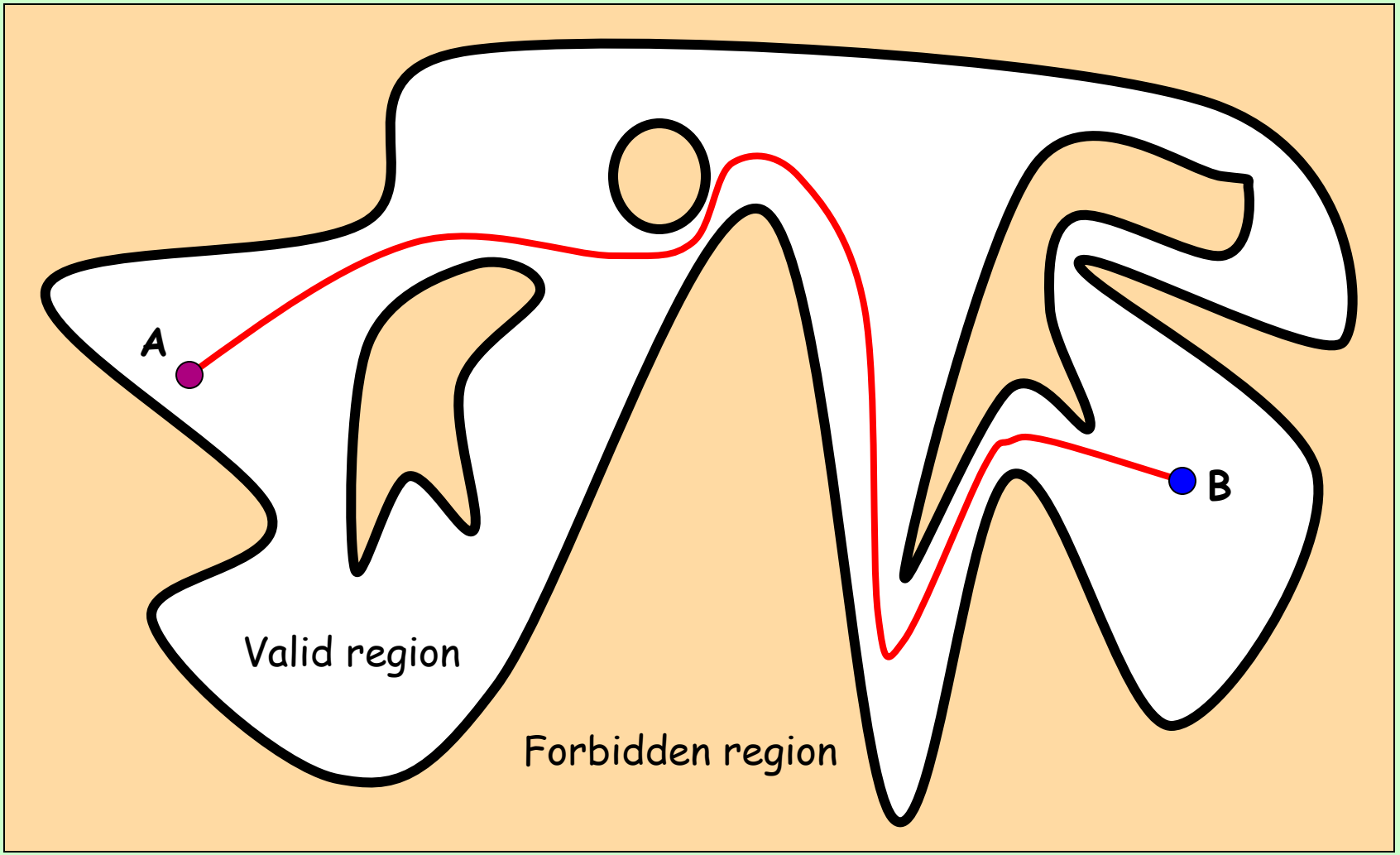


Professor Gurvinder S Virk
Technical Director

InnotecUK Limited
Email: gurvinder.virk@innotecuk.com

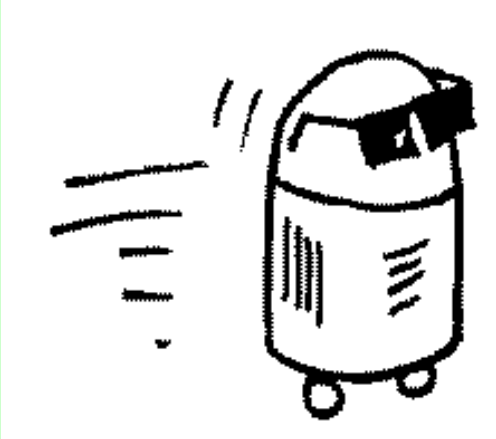
Fundamental question

Are two given points connected by a path?



Localisation and navigation

- **Localisation asks "Where am I?"**
  - **Building a map with an accurate set of sensors – Easy!**
  - **Localization with an accurate map – Simple!**
  - **Fact: You start off with noisy sensors and no map**
- **Navigation provides solution as to where to go (with or without map)!**
  - **Goal directed behaviour**
  - **Obstacle avoidance**



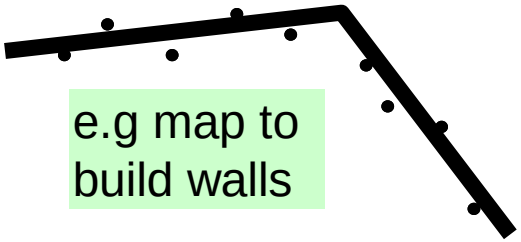
Localisation and mapping

- **Problem 1 – Localisation**
  - **Given**
    - **World map**
    - **Robot's initial pose**
    - **Sensor updates**
  - **Find**
    - **Robot's pose as it moves**
- **Problem 2 – Mapping**
  - **Given**
    - **Robot**
    - **Sensors**
  - **Find**
    - **Map of the environment (and implicitly, the robot's location as it moves)**

What is a Topological map?

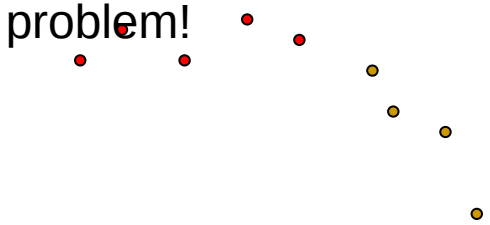
How to build a map?

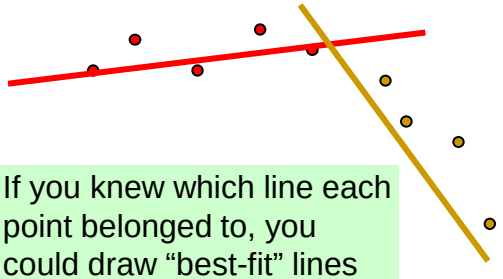
Look at a similar problem in line fitting; Goal: To group a bunch of points into two "best-fit" line segments



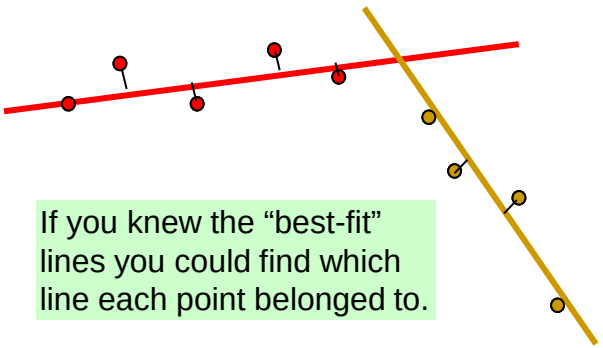
e.g map to build walls

Chicken and egg problem!





If you knew which line each point belonged to, you could draw "best-fit" lines



If you knew the "best-fit" lines you could find which line each point belonged to.

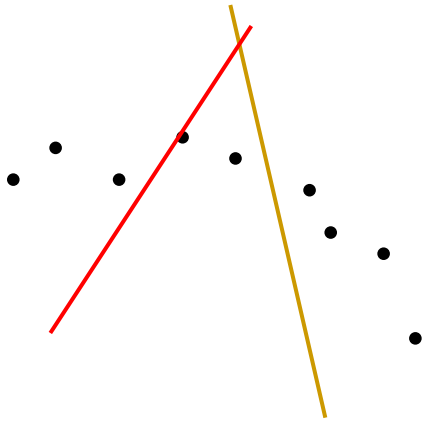
But you don't know either; so how do you proceed?

Algorithm:

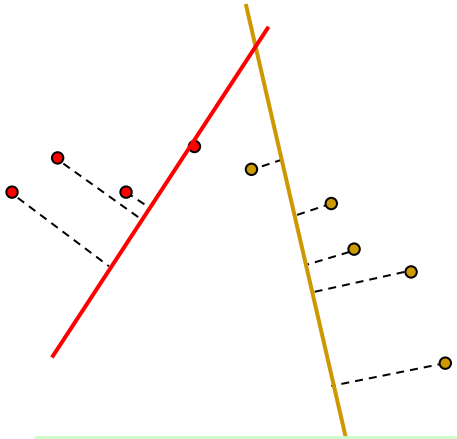
- **Initialize: Make random guess for the lines**
- **Repeat:**
  - **Find the line closest to each point and group into two sets.**
  - **Find the best-fit lines to the two sets**
  - **Iterate until convergence**

- **Algorithm is guaranteed to converge to some local optima (errors are minimised)**

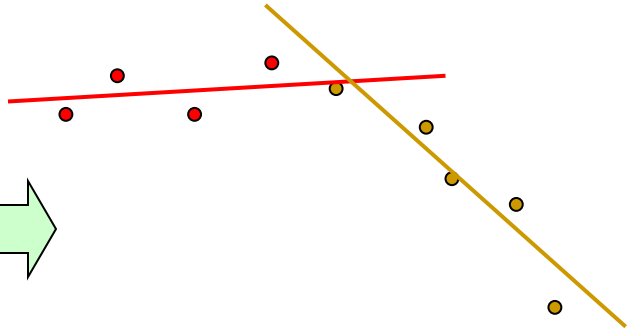
Example



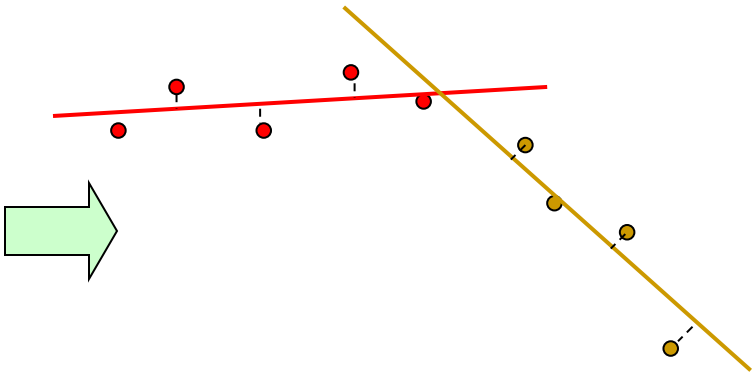
Step 0



Reduce large errors

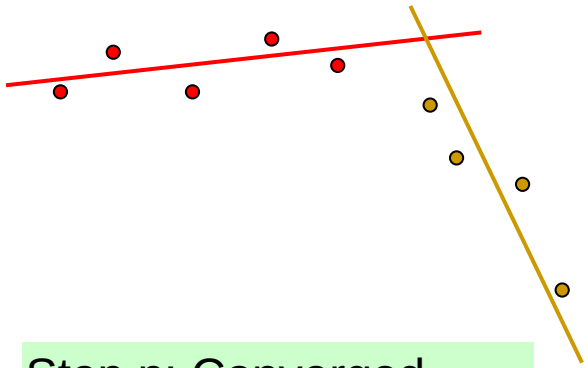


Step 1



Continue reducing the large errors

ETC



Step n: Converged
→ have the map walls

SLAM: Simultaneous *L*ocalization *a*nd *M*apping

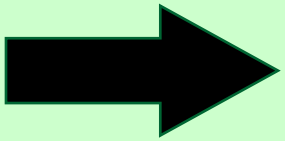
- **SLAM is the problem of determining the location of environmental features by a moving rover**
- **SLAM is a bootstrapping ("chicken and egg") problem:**
  - **to map features in the world, a rover needs to know its own location, but**
  - **to deduce its own location it needs an accurate world map**
- **SLAM applies to any environment: land, air, sea, or space**
- **SLAM needs to be solved for almost any autonomous agent's goal (e.g. navigation, terrain exploration, obstacle avoidance, and camera calibration)**

Approach: Iterated Extended Kalman Filter (IEKF)

old estimate of the true state

noisy measurements of the state

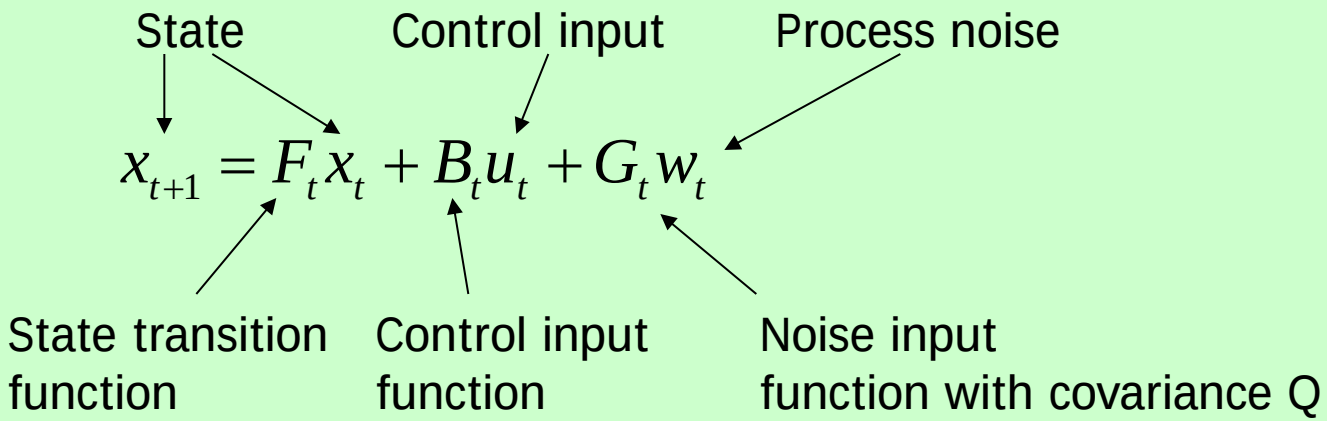
IEKF



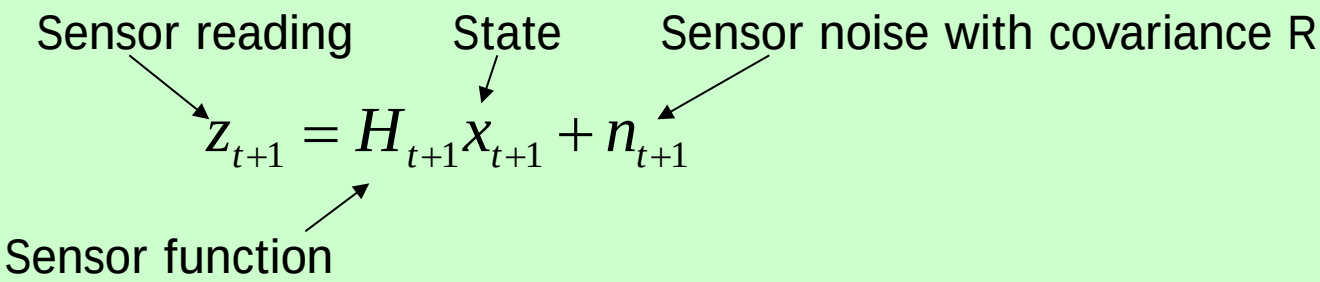
"better" estimate of the true state

Kalman Filter Components

Linear discrete time dynamic system (motion model)



Measurement equation (sensor model)



The Kalman Filter: gives best prediction

Given a set of measurements: $Z_{t+1} = \{z_i, i \le t+1\}$

According to the Fundamental Theorem of Estimation, the state that minimises the mean square error covariance will be:

$$\hat{x}^{MMSE} = E[x \mid Z_{t+1}]$$

$$P^{MMSE} = E[(x - \hat{x})^2 \mid Z_{t+1}]$$

Following notation commonly used:

$$\hat{x}_{t+1|t+1} = E[x_{t+1} \mid Z_{t+1}]$$

$$\hat{x}_{t+1|t} = E[x_{t+1} \mid Z_t]$$

Kalman Filter gives best prediction which improves as more data/ information becomes available:

Classification of Localisation Methods

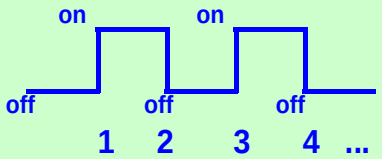
- **Relative**
  - **Odometry: Uses encoders to measure wheel rotation. Is self contained and is ever ready to provide the vehicle with an estimate of position. Position error grows out of bound**
  - **Inertial Navigation: Uses gyroscopes and accelerometers to measure rates of rotation and acceleration. Self contained. Unsuitable for accurate positioning over extended periods of time. High manufacturing and equipment cost**
- **Absolute**
  - **Active Beacons: Computes the absolute position of the robot by measuring the direction of incidence of three or more actively transmitted beacons**
  - **Artificial Landmark Recognition: Distinctive landmarks placed in known locations. Errors are bounded. Computationally intensive and raises questions for persistent real-time position updates**
  - **Map-Based Positioning:**

Localisation: Relative

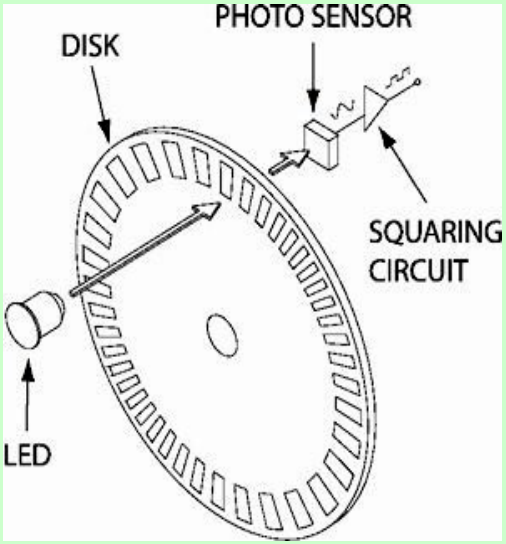
- If you know your speed and direction, you can calculate where you are relative to where you were at start (by integrating your path)
- Speed and direction might, themselves, be absolute (compass, speedometer), or integrated (gyroscope, accelerometer)
- Relative measurements are usually more accurate in the short term - but suffer from accumulated error in the long term
- Most robotics work seems to focus on this

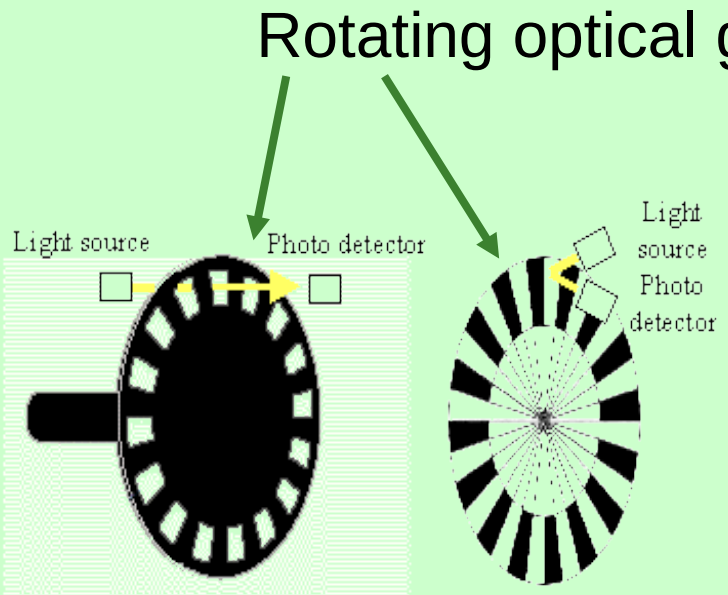
Wheel/ Motor Encoders

- **Used to measure rotational motion ⇒ measure rotation of a wheel**
  - **Servo motors: Used in conjunction with an electric motor to measure the motor's position and, in turn control its position**
- **Principle: Photo detection + optical graduated disk**
- **Direction of motion: Quadrature encoder**
- **Output: Read values with polling or use interrupts**
- **Resolution: 2,000 (⇒10,000) cycles per revolution**
  - **for higher resolution: interpolation, sine waves**
- **Accuracy: no systematic error (accuracy~100%)**



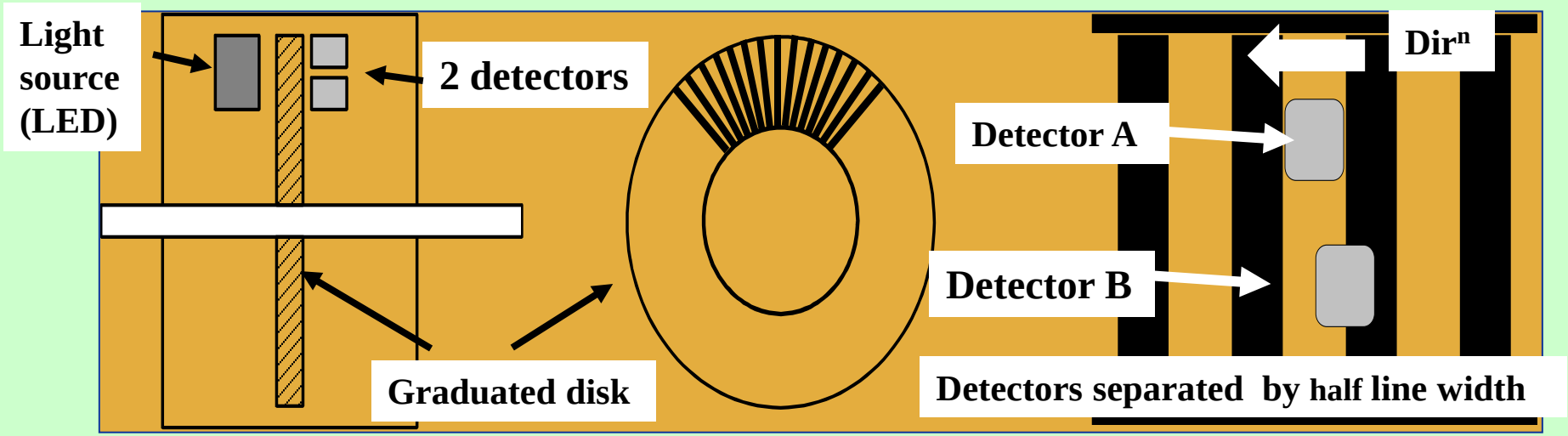
Output is a voltage square wave and important specification is the number of counts per revolution

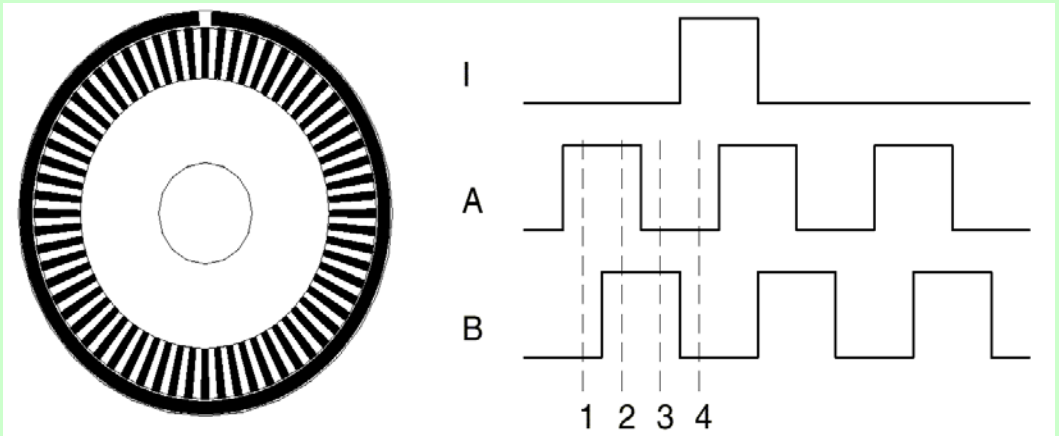




Quadrature wheel/ motor encoders

- The lines interrupt light beam to two photodetectors and get counted. Detectors A and B allow direction of rotation to be determined; High and low going edges for A and B converted to pulses.





State	Ch A	Ch B
S_1	High	Low
S_2	High	High
S_3	Low	High
S_4	Low	Low

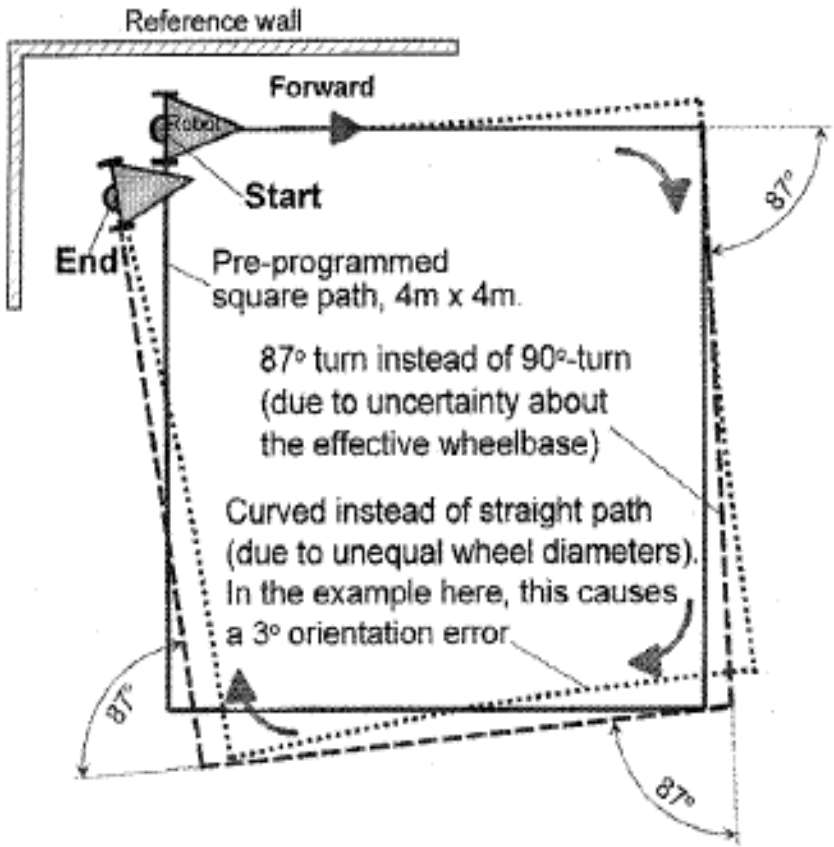
Simple example

- Encoder gives 100 counts/rev
- How far does wheel travel for 1 encoder count?
  - Circum=πD= π *0.1=0.3142=100Δx
  - $\Rightarrow$ Δx=3.142mm
- What happens if we change wheel size?
  - Δx=πD/100.
  - So D$\uparrow$ $\Rightarrow$$\Delta$x$\uparrow$ & D$\downarrow$ Δx$\downarrow$
- If we want 1mm/count, what should wheel diameter be?
  - Circum=10cm $\Rightarrow$ D=10/π=3.3cm

10 cm wheel diameter

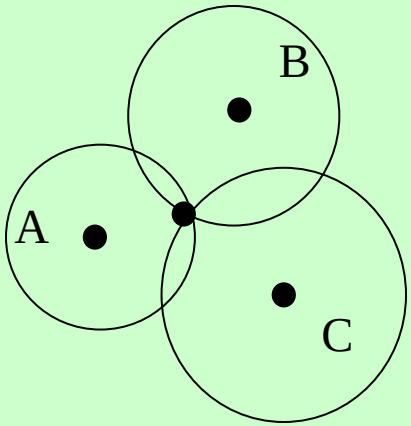
Dead reckoning: accumulation of errors

- **Systematic Errors:**
  - **Unequal wheel diameters**
  - **Average of both wheel diameters differs from nominal diameter**
  - **Misalignment of wheels**
  - **Limited encoder resolution, sampling rate, etc.**
- **Nonsystematic Errors:**
  - **Travel over uneven floors**
  - **Travel over unexpected objects on the floor**
  - **Wheel-slippage due to:**
    - **Slippery floors; over-acceleration, fast turning (skidding), non-point wheel contact with the floor**

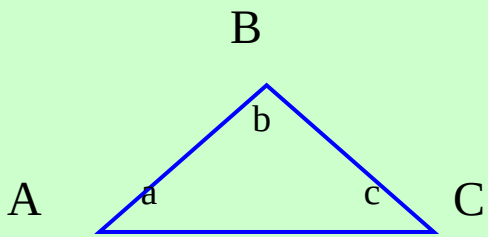


Localisation: Absolute (via beacons)

Trilateration (distances)



Triangulation (angles)



Sines Rule

$$\frac{A}{\sin a} = \frac{B}{\sin b} = \frac{C}{\sin c}$$

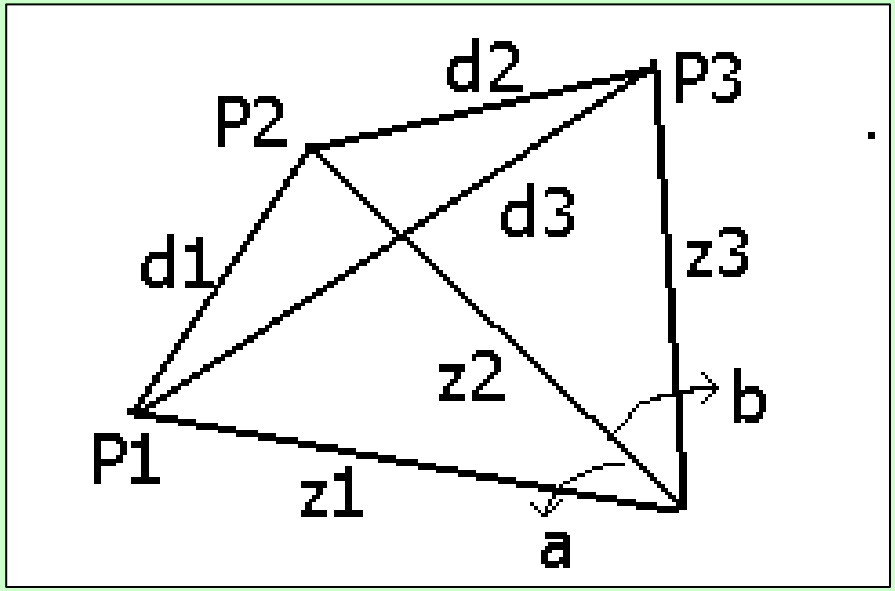
Cosines Rule

$$C^2 = A^2 + B^2 + 2AB\cos(c)$$

$$B^2 = A^2 + C^2 - 2BC\cos(b)$$

$$C^2 = B^2 + C^2 - 2BC\cos(a)$$

Triangulation using Landmarks (angles)



P1, P2 and P3 are known Landmarks

$d1^2 = z1^2 + z2^2 - 2||z1||\,||z2||\cos(a)$

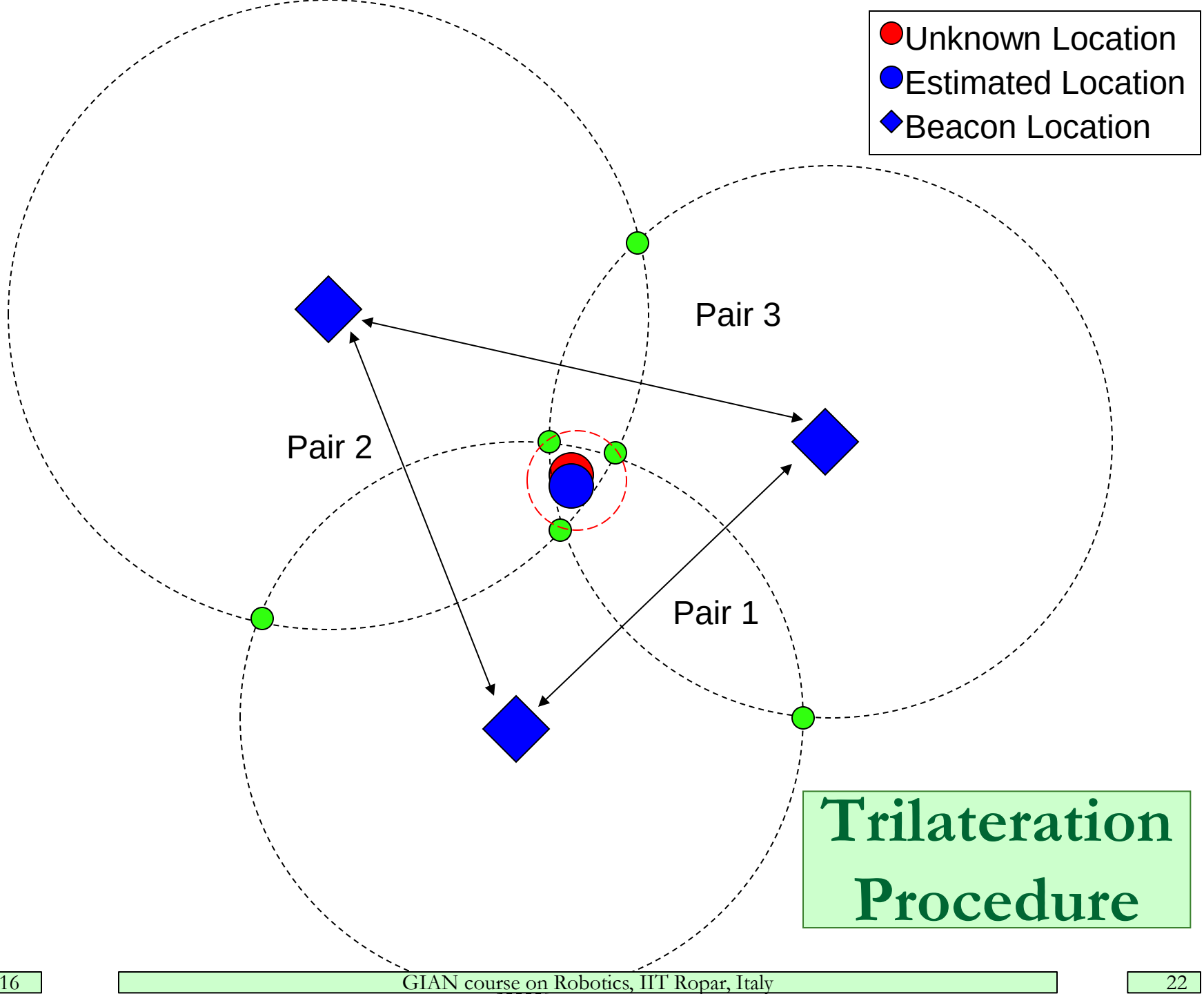
$d2^2 = z2^2 + z3^2 - 2||z1||\,||z2||\cos(b)$

$d3^2 = z1^2 + z3^2 - 2||z1||\,||z3||\cos(a+b)$

Solve for z1, z2 and z3 to obtain robot's position and orientation

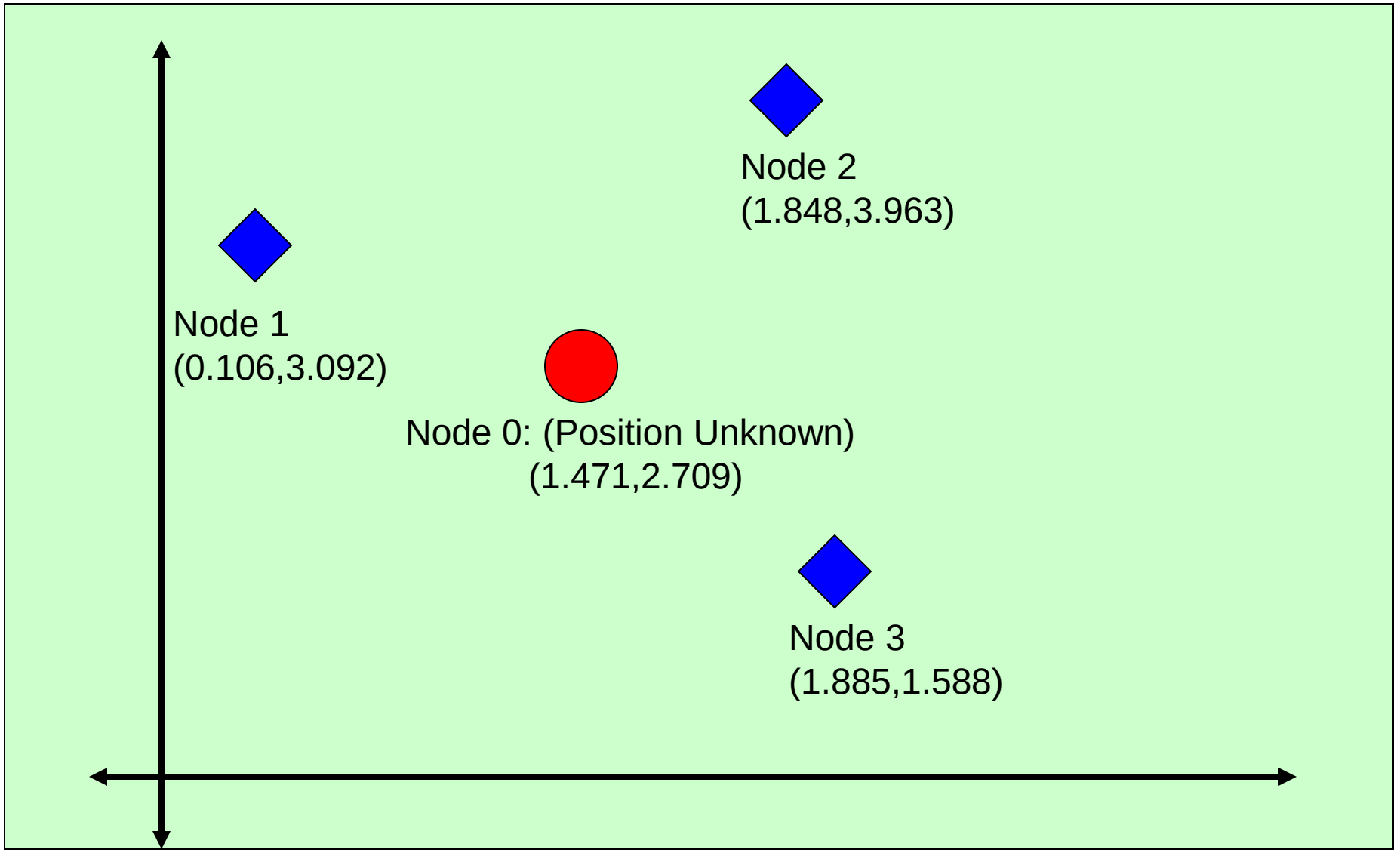
Trilateration algorithm (distances)

- Forms pairs of beacons from set of 3
- Each pair evaluates two possible solutions
  - Forms set of 6 possible locations
- Closest 3 points averaged to get a reasonable solution
- Can be performed using 3 beacons nodes as shown in diagram on next slide



Trilateration Procedure

Trilateration example



Intersection of two circles: centres a and b at (ax, ay) and (bx, by) radii ad and bd respectively

$$(x-a_x)^2+(y-a_y)^2=a_d^2$$

$$(x-b_x)^2+(y-b_y)^2=b_d^2$$

$$x^2-2a_x x+a_x^2+y^2-2a_y y+a_y^2=a_d^2 \qquad (1)$$

$$x^2-2b_x x+b_x^2+y^2-2b_y y+b_y^2=b_d^2 \qquad (2)$$

$$(2)-(1)\Rightarrow$$

$$-2b_x x+2a_x x+b_x^2-a_x^2-2b_y y+2a_y y+b_y^2-a_y^2=b_d^2-a_d^2$$

$$2(a_x-b_x)x+2(a_y-b_y)y=b_d^2-a_d^2-b_x^2+a_x^2-b_y^2+a_y^2$$

$$\Rightarrow c_1x+c_2y=c_3 \quad \Rightarrow \quad x=c_4-c_5y \qquad (3)$$

$$subtitute \quad in \quad (2)$$

$$(c_5^2+1)y^2+(-2c_4c_5+2b_xc_5-2b_y)y+c_4^2-2b_xc_4+b_x^2+b_y^2-b_d^2=0$$

Solve quadratic for y and substitute into (3) to determine x. Then have the intersection points for the two circles. Easily done via Matlab.

Numerical example

- **Determine:**
- **Label position of robot n0, and beacons as n1, n2 and n3**
- **Determine distances to beacons as d1, d2 and d3**
- **Use formula on previous slide to determine intersection points for combinations of two beacons:**
  - **Beacons n1 & n2: (0.68186, 4.4138) & (1.4710, 2.709)**
  - **Beacons n1 & n3: (0.7107, 1.8097) & (1.4710, 2.709)**
  - **Beacons n1 & n2: (2.2639, 2.7214) & (1.4710, 2.709)**
- **Normally have errors in the measurements; hence locations will have a variance. Choose the roots that are clustered and do some averaging to get best location**

GPS: Global Positioning System

- **24 satellites launched by US DOD, originally for weapons systems targeting; “man-made stars" as reference points to calculate positions accurate to a matter of meters. Now many more**
- **Work started in 1973, testing 1978-94**
- **After $12B → GPS**
- **Gives time & position anywhere in the world, although often only outdoors; it's like giving every square metre on the planet a unique address!**
- **Typical Position Accuracy: 5m but can be much better**
- **Others: European Union Galileo positioning system, China's BeiDou Navigation Satellite System, Japanese Quasi-Zenith Satellite System, and India's Indian Regional Navigation Satellite System (NAVIC)**

Basic idea

- Satellites constantly transmit beacons along with the time-of-beacon and position (in predictable, corrected, and observed orbits)
- Receivers listen for (phase-shifted) signals and compute distance based on propagation delay
- 3 satellites gives you 2 points (in 3D); throw out (the one in deep space)
- Compute position relative to satellites; use satellite position to get Earth coordinates

Introduction to Navigation

1. **What is the aim of navigation? → Go from Point A to Point B**
   - **Where am I? → Localisation**
   - **Where is the goal with respect to me?**
   - **How do I get there from here?**
   - **Usually don't want collisions**
2. **How is navigation actually achieved? → by locating with respect to a map and inferring a navigable path. This involves**
   - **Sensing activities (localisation)**
   - **Analyses (some form of thinking)**
   - **Moving (carrying out the decision taken)**
3. **Problems needing addressing**
   - **Knowledge of the operating environment (a map)**
   - **Dynamics of the environment (are things moving about? How?)**
   - **Learn from biology? → Good strategies exist in nature**

Types of Navigation

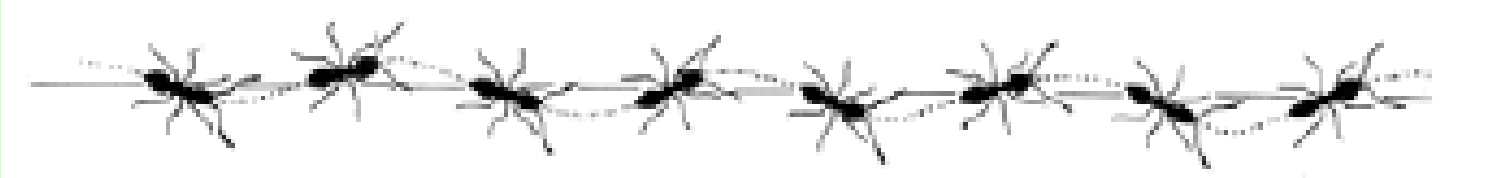
- **Searching: can move via any strategy (even randomly), and can recognise arrival at goal**
- **Direction following: eg compass direction or trail following; can find goal from one direction**
- **Aimed navigation: eg. Taxis to source, using landmarks (with pointers); can find a salient goal from a catchment area**
- **Guidance by surroundings: recognition triggered response; topological route integration; and survey (map needed)**

Navigational strategies in animals

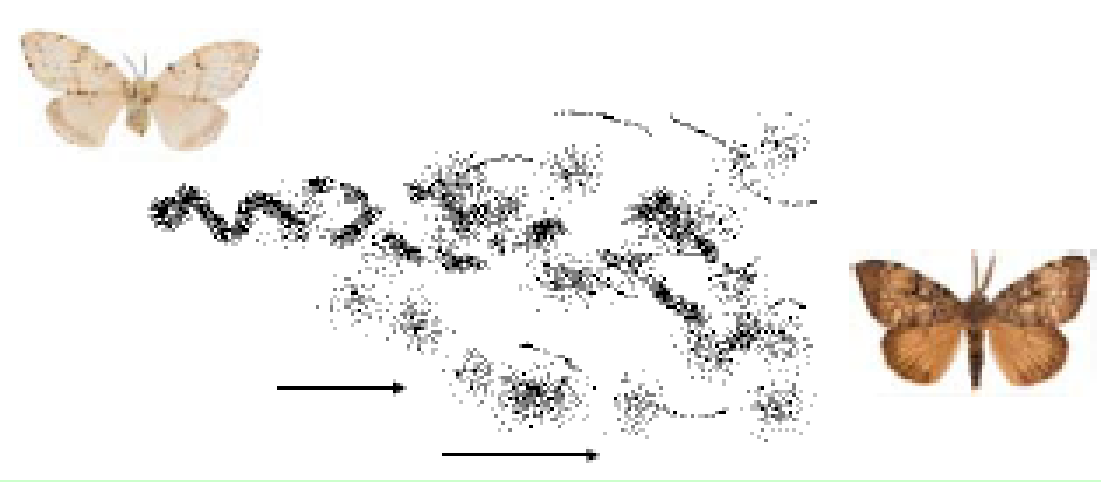
- **Orientation behaviour: How organisms respond to an external stimuli**
  - **Orientation: turning response, or oriented response guided by stimulus location or direction**
  - **Response selection: choice of response behaviour as appropriate to stimulus**
- **Nearly every behaviour has a component to it that involves orientation**
- **Examples of oriented behaviour**
  - **Chemotaxis; pheromones**
  - **Bioluminescence (fire flies)**
  - **Chemical trail following**
  - **Directional tactile sense**
  - **Using landmarks and memory**

Examples of oriented behaviour

- **Trail following in an ant**



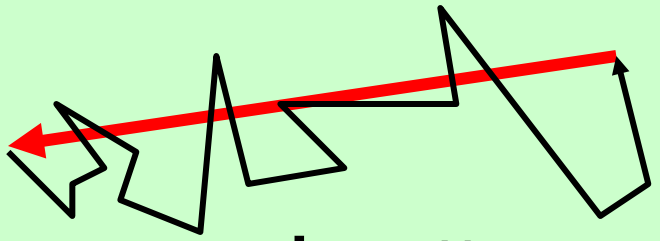
- **Olfactory orientation to prey by a snake**
- **Pheromone sensing in moths**



- **Orientation to vibration sources**

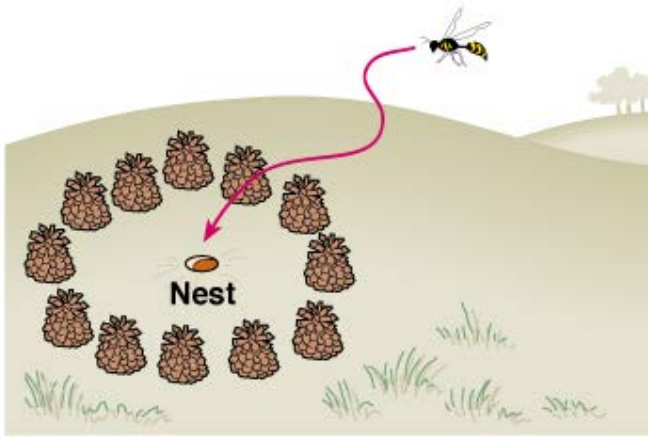
Taxis strategies in animals

- **Taxis: a directed movement towards (or away from) a stimulus**
  - **Phototaxis (light)**
  - **Geotaxis (gravity)**
  - **Phonotaxis (sound)**
  - **Chemotaxis (chemicals)**

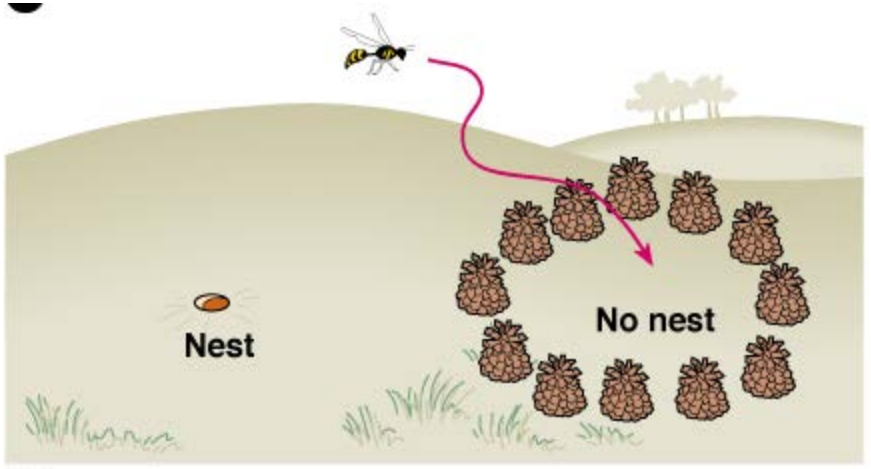


- **Desert ants (Cataglyphis): random search pattern + straight run to home (visual receptors tuned to the polarisation plane of light – skylight has a natural polarisation pattern)**
- **Kitchen ants: random search pattern until they find trails left by other ants**

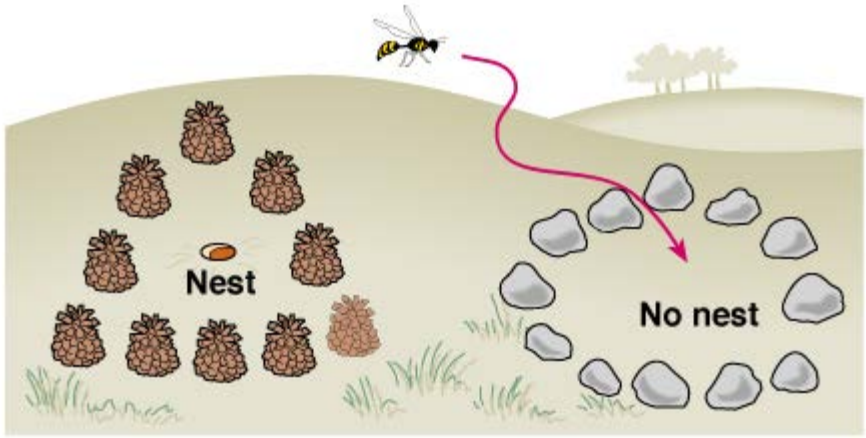
Orientation in Digger Wasps



1



2



3

Digger wasp leaves nest and circles opening for 6 seconds. Pine cones are places there during time when she is below ground.

It is not the pine cones but the shape of the landmarks that are important. On return the wasp goes to where it remembers the nest to be for both cases (2) and (3).

The fitness is enhanced by the wasp's ability to find the nest

Navigational Methods: Artificial

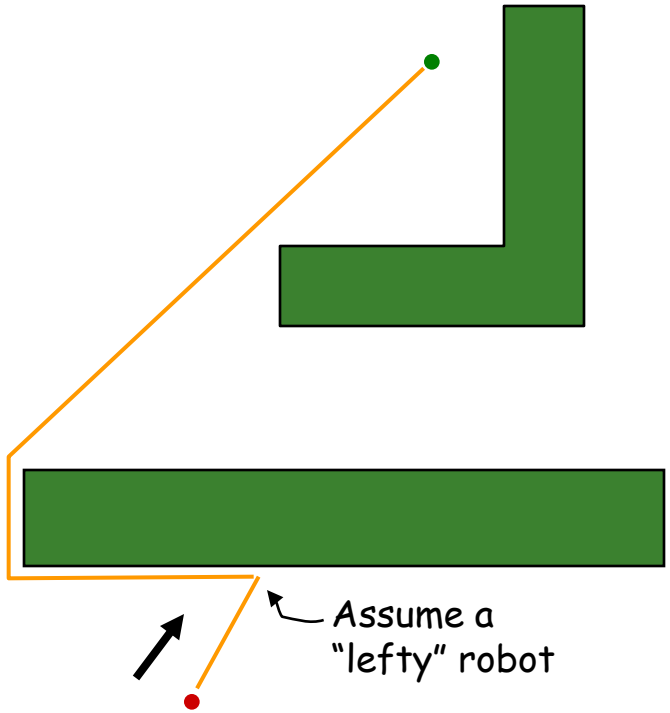
Man-made navigation methods

- **Classical methods: Runaway, wander, wall following, obstacle avoidance, follow-the-leader, subsumption, potential fields**
- **AI methods: Expert systems (rule-based), Soft computing (fuzzy logic, neural networks, evolutionary algorithms), hybrid mixing of different techniques**
- **Static environments: use of maps, dead reckoning and/or beacons (landmarks)**
- **Dynamic environments: Same as static case but need to continually update for changing situations**

Need for simple, effective, flexible and robust methods. Biological methods (in the main) tend to satisfy these requirements but man-made methods do not. Lots to do still!!

Simple insect inspired navigation strategy

Insect-inspired “bug” algorithms



- Known direction to goal
- Otherwise only local sensing walls/obstacles encoders

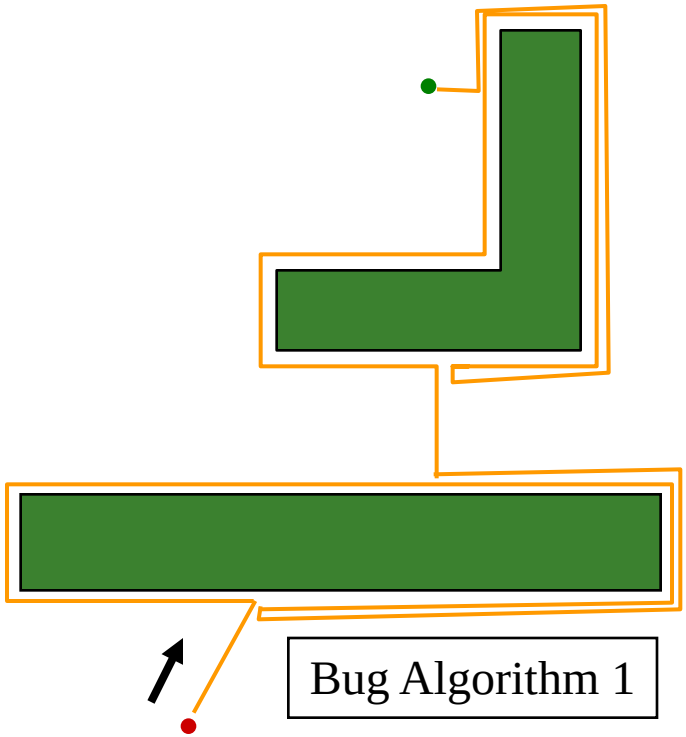
Bug Algorithm 0

1) Head toward goal
2) Follow obstacles until you can head toward the goal again
3) Continue

Problems if robot can’t head towards robot in cluttered situations

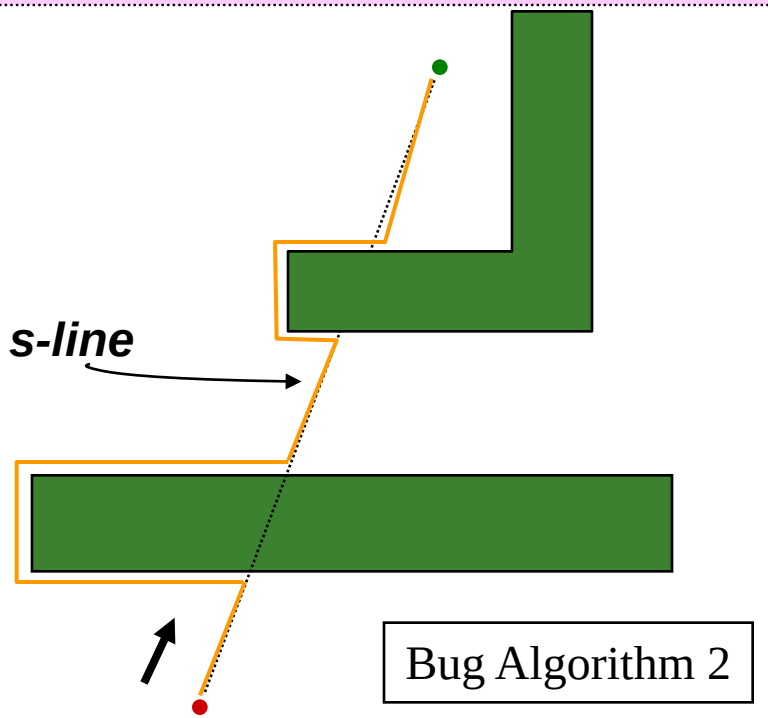
Improved insect algorithms

1) Head toward goal
2) If an obstacle is encountered, circumnavigate it *and* remember how close you get to the goal
3) Return to that closest point (by wall-following) and continue



Bug Algorithm 1

1) Head toward goal on the *s-line*
2) If an obstacle is in the way, follow it until encountering the s-line again *closer to the goal*.
3) Leave the obstacle and continue toward the goal

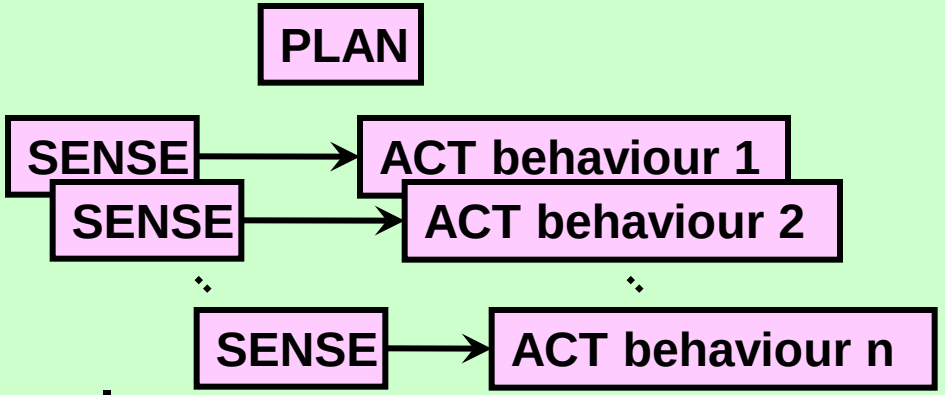


Bug Algorithm 2

Robot navigational strategies

- **Sense, plan, act: processing times can be long for real-time applications**

- **Behavioural based (subsumption architecture, Brooks, 1986)**
  - **Several layers of circuitry (behaviours) which have close coupling between the sensing and acting. Very Fast response**
  - **Each layer is functionally equivalent to a simple whole system, e.g.**
    - **Obstacle avoidance**
    - **Exploration**
    - **Recognition**
    - **Layers work in parallel**
    - **Simple interaction between layers**



Summary

- Introduction to the robot motion planning
- Localisation and navigation overview
- Examples provided

Thanks are expressed to all the colleagues (too many to mention individually) across the world who have helped with the formulation of this lecture on Localisation and navigation from material placed on the www