Robotics

Lecture 4: Probabilistic Robotics

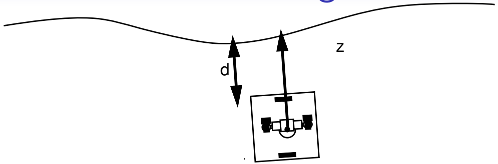
See course website
http://www.doc.ic.ac.uk/~ajd/Robotics/ for up to
date information.

Andrew Davison
Department of Computing
Imperial College London

Review: Sensors Practical from Lecture 3

- Touch sensor (returns yes/no state): use to detect collision and trigger avoidance action.
- Sonar sensor (returns depth value in cm): can be used for smooth *servoing* behaviour with proportional gain.
- Both are examples of negative feedback.

Review: Wall Following with Sonar



- Use sideways-looking sonar to measure distance z to wall.
- Use velocity control and a while loop at for instance 20Hz.
- With the goal of maintaining a desired distance d, set difference between left and right wheel velocities proportional to difference between z and d:

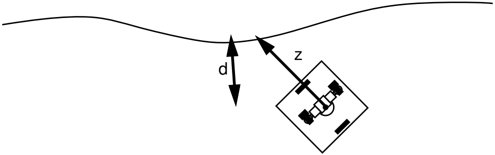
$$v_R - v_L = K_p(z - d)$$

Symmetric behaviour can therefore be achieved using a constant offset v_C:

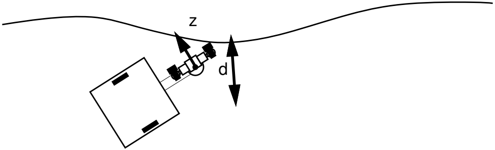
$$v_R = v_C + \frac{1}{2}K_p(z - d)$$

$$v_L = v_C - \frac{1}{2}K_p(z - d)$$

Review: Wall Following with Sonar

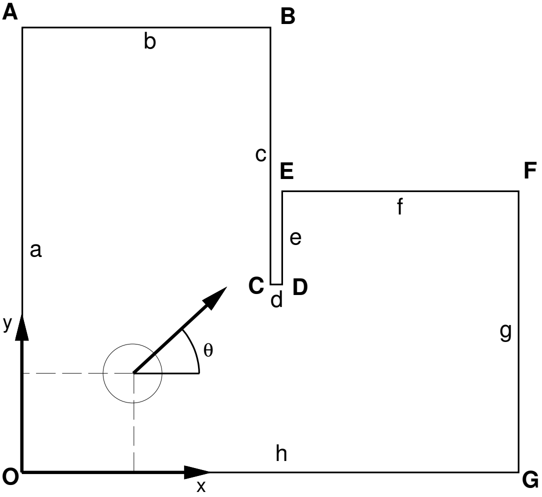


- Problem if angle between robot's direction and wall gets too large because sonar doesn't measure the perpendicular distance.
- Solutions: ring of sonar sensors would be most straightforward. Clever combination of measurements from different times?
- Better result with sonar mounted forward of wheels.



Probabilistic Localisation

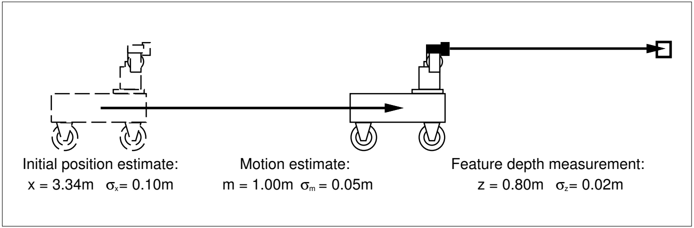
Over the next two weeks we will aim at much more reliable navigation: possible if the robot actually *knows where it is* relative to a map.



Probabilistic Robotics

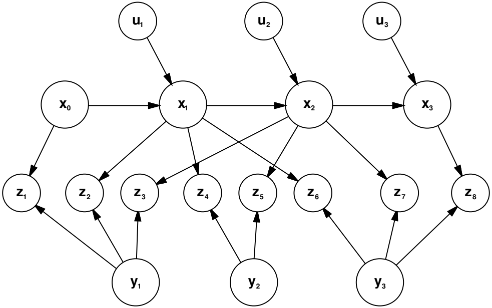
- Problem: simple sensing/action procedures can be locally effective but are limited in complicated problems in the real-world.
- ‘Classical AI’ approaches based on logical reasoning about true/false statements fall down when presented with real-world data.
- Why?
  - Advanced sensors don’t lend themselves to straightforward analysis like bump and light sensors.
  - All information which a robot receives is uncertain.
- A probabilistic approach acknowledges uncertainty and uses models to abstract useful information from data.

Uncertainty in Robotics



- Every robot action is uncertain.
- Every sensor measurement is uncertain.
- When we combine actions and measurements and want to estimate the state of a robot, the state estimate will be uncertain.

Probabilistic Inference



- What is my state and that of the world around me?
- Prior knowledge is combined with new measurements.
- A series of weighted combinations of old and new information.
- Sensor fusion: the general process of combining data from many different sources into useful estimates.
- This composite state estimate can then be used to decide on the robot's next action.

Bayesian Probabilistic Inference

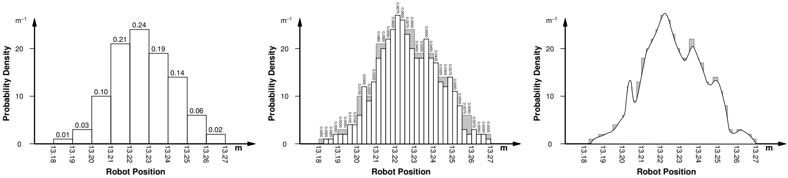
- 'Bayesian' has come to be known as a certain view of the meaning of probability theory as a measure of subjective belief.
- Probabilities describe our state of knowledge — nothing to do with randomness in the world.
- Bayes' Rule relating probabilities of discrete statements:

$$\begin{aligned} P(\mathbf{XZ}) &= P(\mathbf{Z}|\mathbf{X})P(\mathbf{X}) = P(\mathbf{X}|\mathbf{Z})P(\mathbf{Z}) \\ \Rightarrow P(\mathbf{X}|\mathbf{Z}) &= \frac{P(\mathbf{Z}|\mathbf{X})P(\mathbf{X})}{P(\mathbf{Z})} \end{aligned}$$

- Here $P(\mathbf{X})$ is the prior; $P(\mathbf{Z}|\mathbf{X})$ the likelihood; $P(\mathbf{X}|\mathbf{Z})$ the posterior; $P(\mathbf{Z})$ sometimes called marginal likelihood.
- We use Bayes's Rule to incrementally digest new information from sensors about a robot's state. Straightforward use for discrete inference where $\mathbf{X}$ and $\mathbf{Z}$ each have values which are one of several labels such as the identity of the room a robot is in.

Probability Distributions: Discrete and Continuous

- Discrete probabilistic inference generalises to large numbers of possible states as we make the bin size smaller and smaller.
- A continuous Probability Density Function $p(x)$ is the limiting case as the widths of bins in a discrete histogram tend to zero.

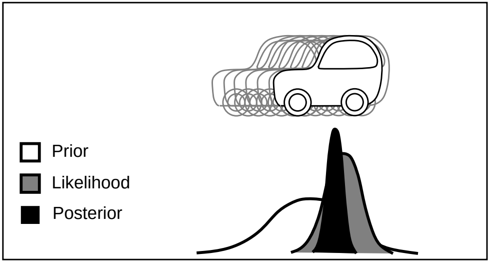


- The probability that a continuous parameter lies in the range a to b is given by the area under the curve:

$$P_{a\rightarrow b} \int_a^b p(x)dx$$

- But generic high resolution representation of probability density is very expensive in terms of memory and computation.

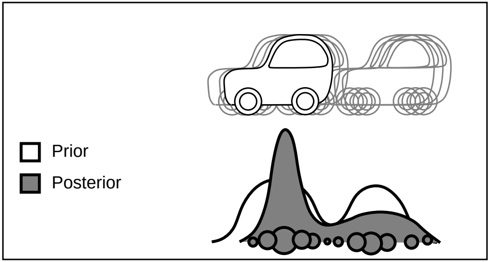
Probability Representations: Gaussians



$$p(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- Explicit Gaussian (or normal) distributions are often represent the uncertainty in sensor measurements very well.
- Wide Gaussian *prior* multiplied by *likelihood* curve to produce a *posterior* which is tighter than either.

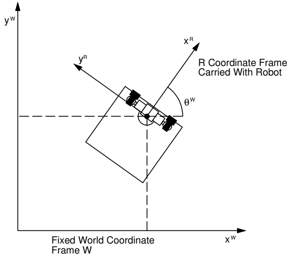
Probability Representations: Particles

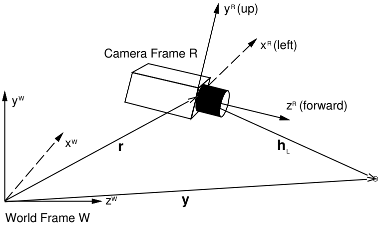


- Here a probability distribution is represented by a finite set of *weighted samples* of the state $\{\mathbf{x}_i, w_i\}$, where $\sum_i w_i = 1$.
- Big advantage is the ability to represent *multi-modal* distributions (with more than one peak) in ambigiuous situations.
- Poor ability to represent detailed shape of distribution when number of particles is low.

Probabilistic Localisation

- The robot has a map of its environment in advance.
- The only uncertain thing is the position of the robot.

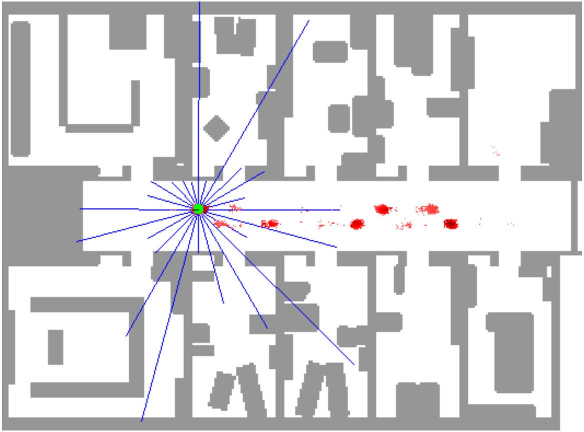




- The robot stores and updates a *probability distribution* representing its uncertain position estimate.

Monte Carlo Localisation (Particle Filter)

- Cloud of particles represent uncertain robot state: more particles in a region = more probability that the robot is there.



(Dieter Fox *et al.*1999, using sonar. See animated gif at `http://www.doc.ic.ac.uk/~ajd/Robotics/RoboticsResources/montecarlolocalization.gif` .)

The Particle Distribution

- A particle is a point estimate $\mathbf{x}_i$ of the state (position) of the robot with a weight w_i.

$$\mathbf{x}_i = \begin{pmatrix} x_i \\ y_i \\ \theta_i \end{pmatrix}$$

- The full particle set is:

$$\{\mathbf{x}_i, w_i\} \ ,$$

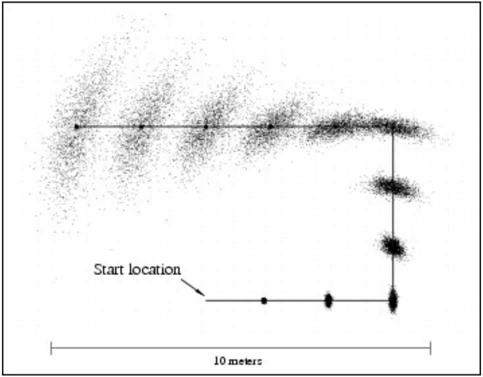
for $i = 1$ to N. A typical value might be $N = 100$.

- All weights should add up to 1. If so, the distribution is said to be *normalised*:

$$\sum_{i=1}^{N} w_i = 1 \ .$$

Displaying a Particle Set

- We can visualise the particle set by plotting the x and y coordinates as a set of dots; more difficult to visualise the θ angular distribution (perhaps with arrows?) — but we can get the main idea just from the linear components.

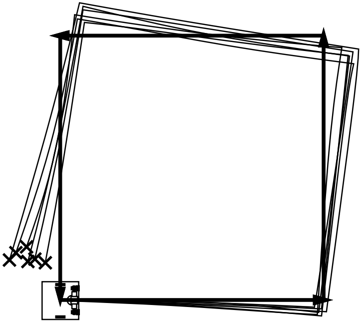


Steps in Particle Filtering

These steps are repeated every time the robot moves a little and makes measurements:

1. Motion Prediction based on Proprioceptive Sensors.
2. Measurement Update based on Outward-Looking Sensors.
3. Normalisation.
4. Resampling.

Motion Prediction



- We know that uncertainty grows during blind motion.
- So when the robot makes a movement, the particle distribution needs to shift its mean position but also spread out.
- We achieve this by passing the state part of each particle through a function which has a deterministic component and a random component.

Motion Prediction

- During a straight-line period of motion of distance D:

$$\begin{pmatrix} x_{new} \\ y_{new} \\ \theta_{new} \end{pmatrix} = \begin{pmatrix} x + (D+e)\cos\theta \\ y + (D+e)\sin\theta \\ \theta + f \end{pmatrix}$$

- During a pure rotation of angle angle α:

$$\begin{pmatrix} x_{new} \\ y_{new} \\ \theta_{new} \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta + \alpha + g \end{pmatrix}$$

- Here e, f and g are zero mean *noise* terms — i.e. random numbers typically with a Gaussian distribution. We generate a different samples for each particle, which causes the particles to spread out.
- Watch out for angular wrap-around — i.e. make sure that θ values are always in the range 0° to 360°.

Measurement Updates

- A measurement update consists of applying Bayes Rule to each particle; remember:

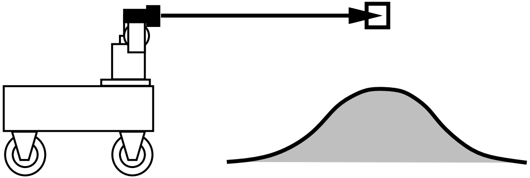
$$P(\mathbf{X}|\mathbf{Z}) = \frac{P(\mathbf{Z}|\mathbf{X})P(\mathbf{X})}{P(\mathbf{Z})}$$

- So when we achieve a measurement z, we update the weight of each particle as follows:

$$w_{i(new)} = P(z|\mathbf{x}_i) \times w_i \ ,$$

  remembering that the denominator in Bayes' rule is a constant factor which we do not need to calculate because it will later be removed by normalisation.
- $P(z|\mathbf{x}_i)$ is the *likelihood* of particle i; the probability of getting measurement z given that it represents the true state.

Likelihood Function



- The form of a likelihood function comes from a probabilistic model of the outward-looking sensor.
- Having calibrated a sensor and understood the uncertainty in its measurements we can build a probabilistic measurement model for how it works. This will be a probability distribution (specifically a likelihood function) of the form:

$$P(z|\mathbf{x}_i)$$

Such a distribution will often have a Gaussian shape.