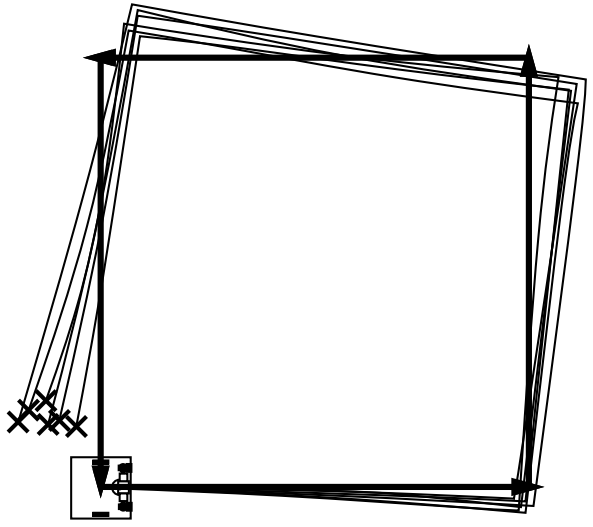
Robotics

Lecture 4: Probabilistic Robotics

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

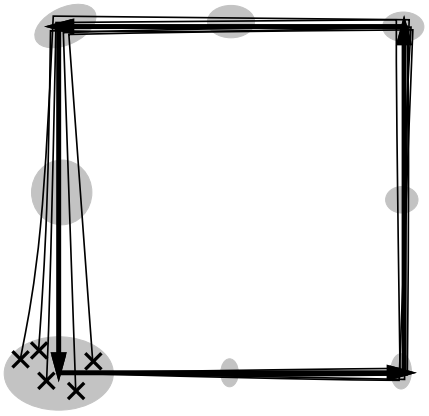
Andrew Davison
Department of Computing
Imperial College London

Review: Practical 1



- Gradual 'motion drift' from perfect square
- Causes: initial alignment errors; wheel slip; miscalibration; unequal left/right motors; others . . . ?
- Careful calibration of distance and angle (probably in position control, by adjusting demand in degrees, and also possibly maximum speed and gains) improves matters but we will never achieve a perfect result every time in this experiment.

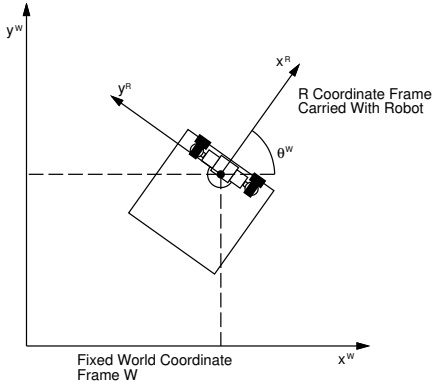
A Well-Calibrated Robot



- After careful calibration the robot should *on average* return to the desired location, but scatter will remain due to uncontrollable factors (variable wheel slip, rough surface, air currents!?. . .)
- *Systematic error* removed; what remains are *zero mean errors*.
- The errors occur *incrementally*: every small additional movement or rotation induces a little more potential error.
- The size of the distribution of the errors in the world frame will grow as the robot moves further around the square.
- We can model the zero mean errors probabilistically: in many cases a Gaussian (normal) distribution is suitable.

What does this mean for estimating motion?

- Perfect motion integration from odometry is not possible:



Recall from the last lecture: state update equations:

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

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

- During a pure rotation of angle angle α:

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

Uncertainty in Motion

- A better model acknowledges that this 'ideal' trajectory is affected by uncertain perturbations ('motion noise'). For example, we could use this simple model:
- During a straight-line period of motion of distance D:

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

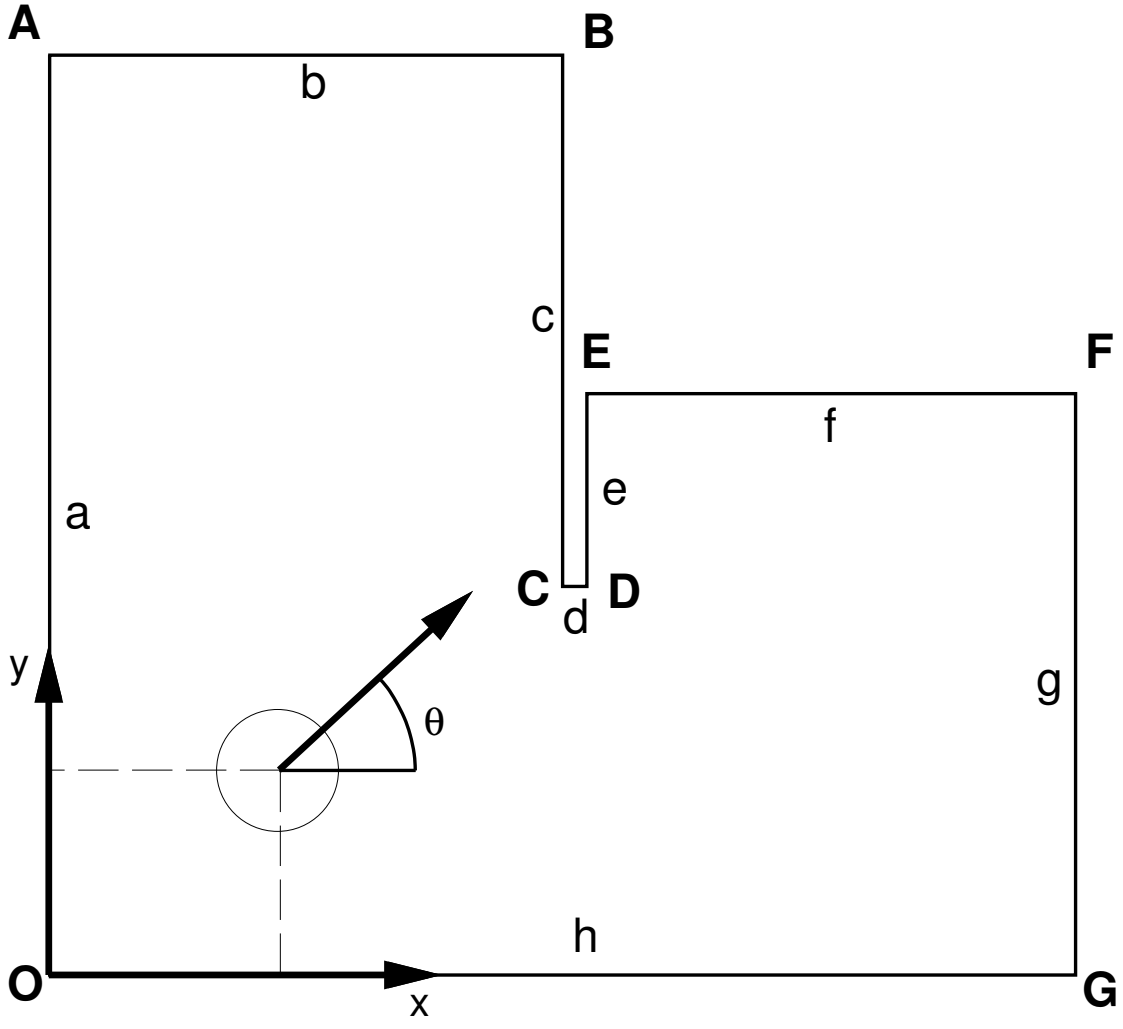
- During a pure rotation of angle α:

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

- Here e, f and g are 'uncertainty' terms, with zero mean and a Gaussian distribution, which model how actual motion might deviate from the ideal trajectory.
- Adding these terms won't help us to move a robot more accurately when it is guided with only odometry; but are important later when we probabilistically combine odometry with other sensing.

Probabilistic Localisation

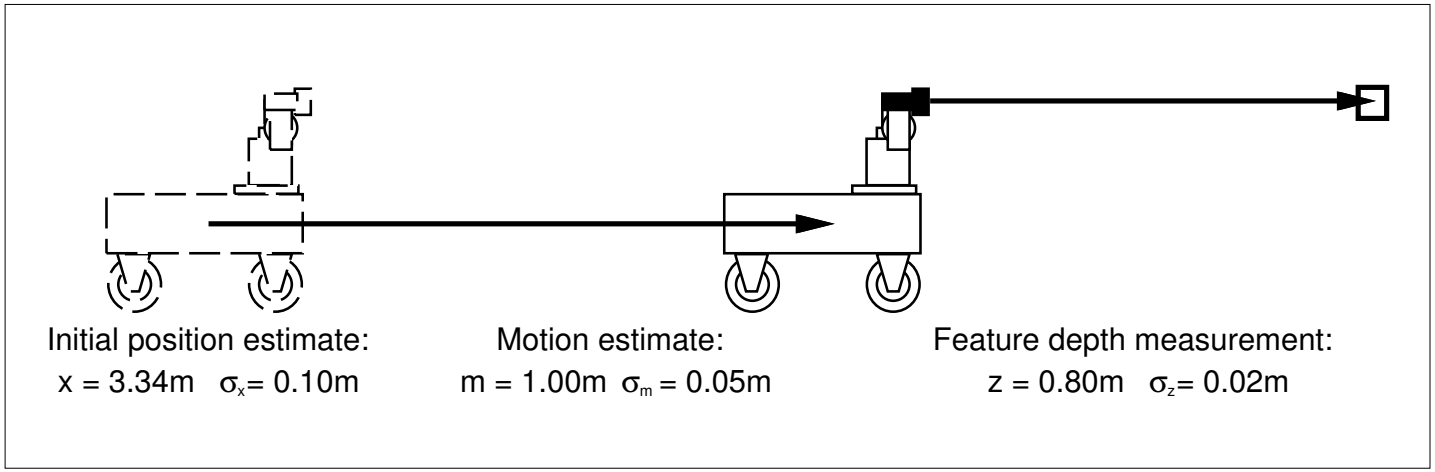
Over the next two weeks we will aim beyond local reactive behaviours towards reliable long-term navigation, via *localisation* relative to a map.



Probabilistic Robotics

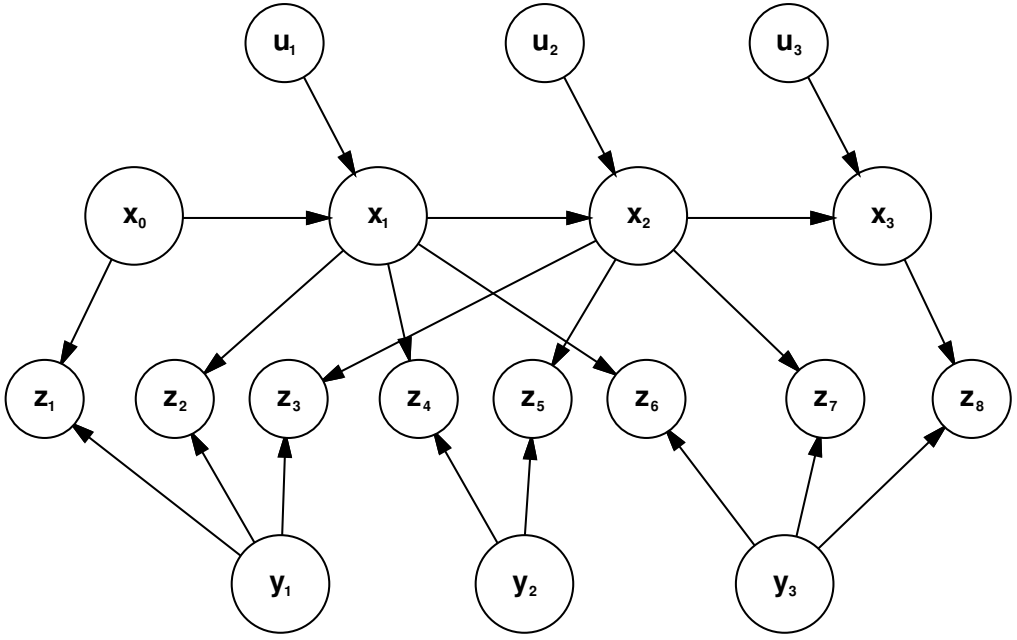
- Problem: simple sensing/action procedures can be locally effective but are limited in complicated problems in the real-world; we need longer-term representations and consistent scene models.
- 'Classical AI' approaches to long-term estimation based on logical reasoning about true/false statements fall 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.
- Our goal is an incrementally updated probabilistic estimate of the position of the robot relative to the map.

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.
- Usually, we will start with some uncertain estimate of the state of a robot, and then take some action and receive new information from sensors. We need to update the uncertain state estimate in response to this.

Probabilistic Inference



- What is my state and that of the world around me?
- Prior knowledge is combined with new measurements; most generally modelled as a Bayesian Network.
- 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

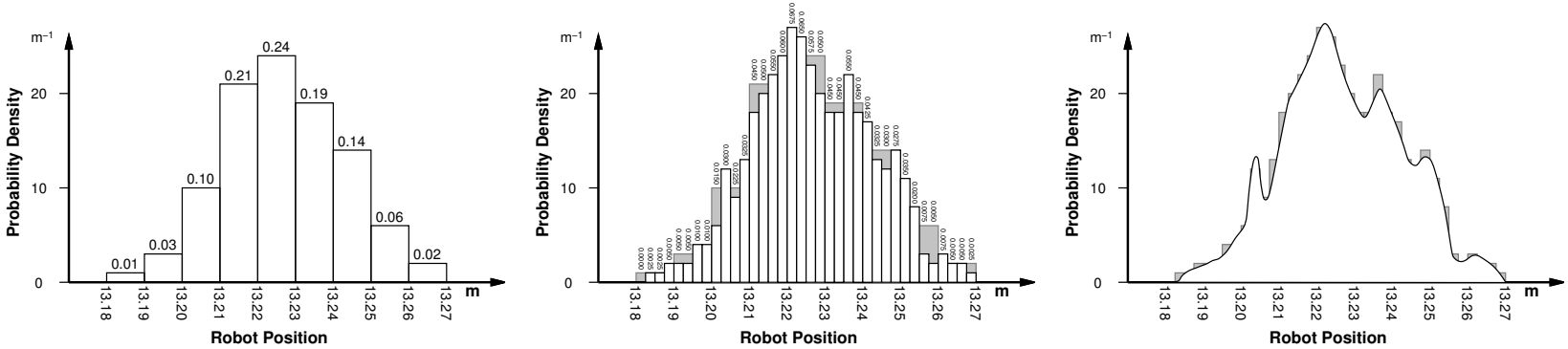
- 'Bayesian' has come to be known as a certain view of the meaning of probability theory as a measure of subjective belief.
- Probabilities describe our state of knowledge — nothing to do with randomness in the world.
- Bayes' Rule relating probabilities of discrete statements:

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

- 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.

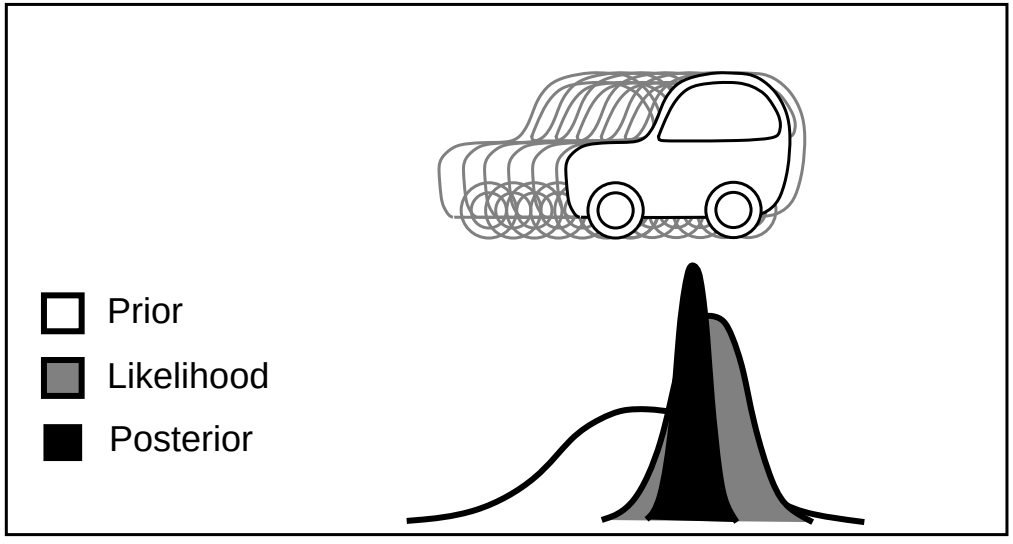


- The probability that a continuous parameter lies in the range a to b is given by the area under the curve:

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

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

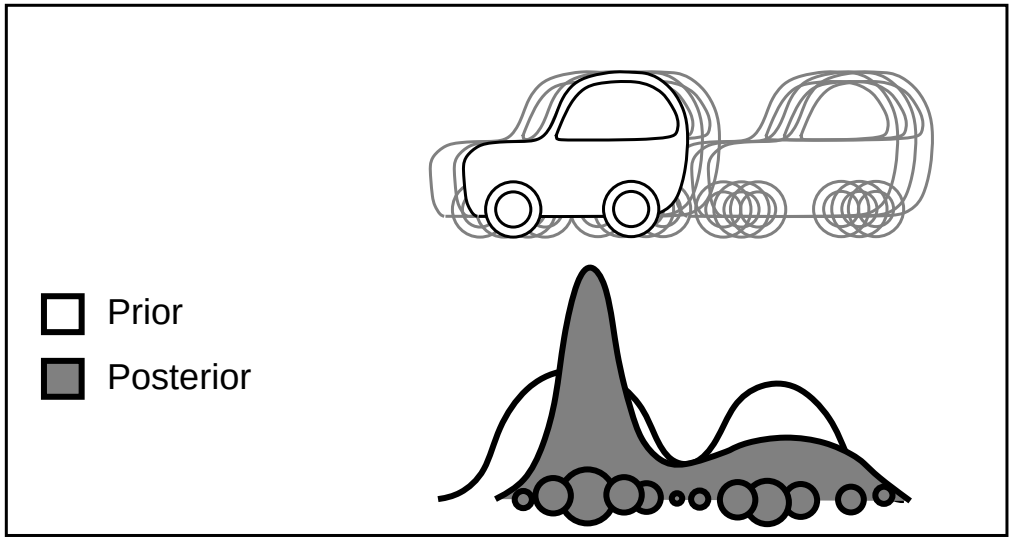
Probability Representations: Gaussians



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

- 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. The product of two Gaussians is always another Gaussian.

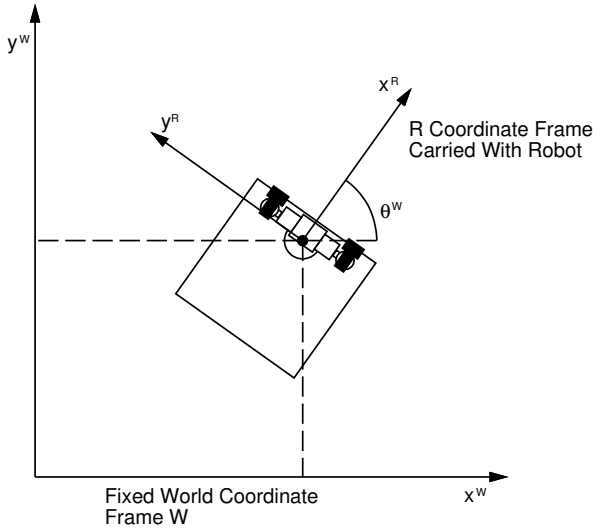
Probability Representations: Particles

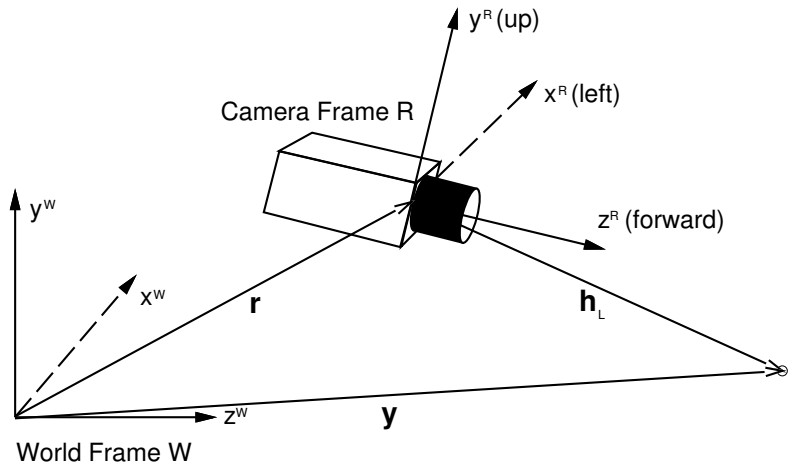


- Here a probability distribution is represented by a finite set of *weighted samples* of the state $\{\mathbf{x}_i, w_i\}$, where $\sum_i w_i = 1$.
- Big advantages are simplicity and the ability to represent and shape of distribution, including *multi-modal* distributions (with more than one peak) in ambigiuous situations.
- Disadvantages are a poor ability to represent detailed shape of distribution when number of particles is low. If we increase the number of particles to improve this, the computational cost can be very high.

Probabilistic Localisation

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

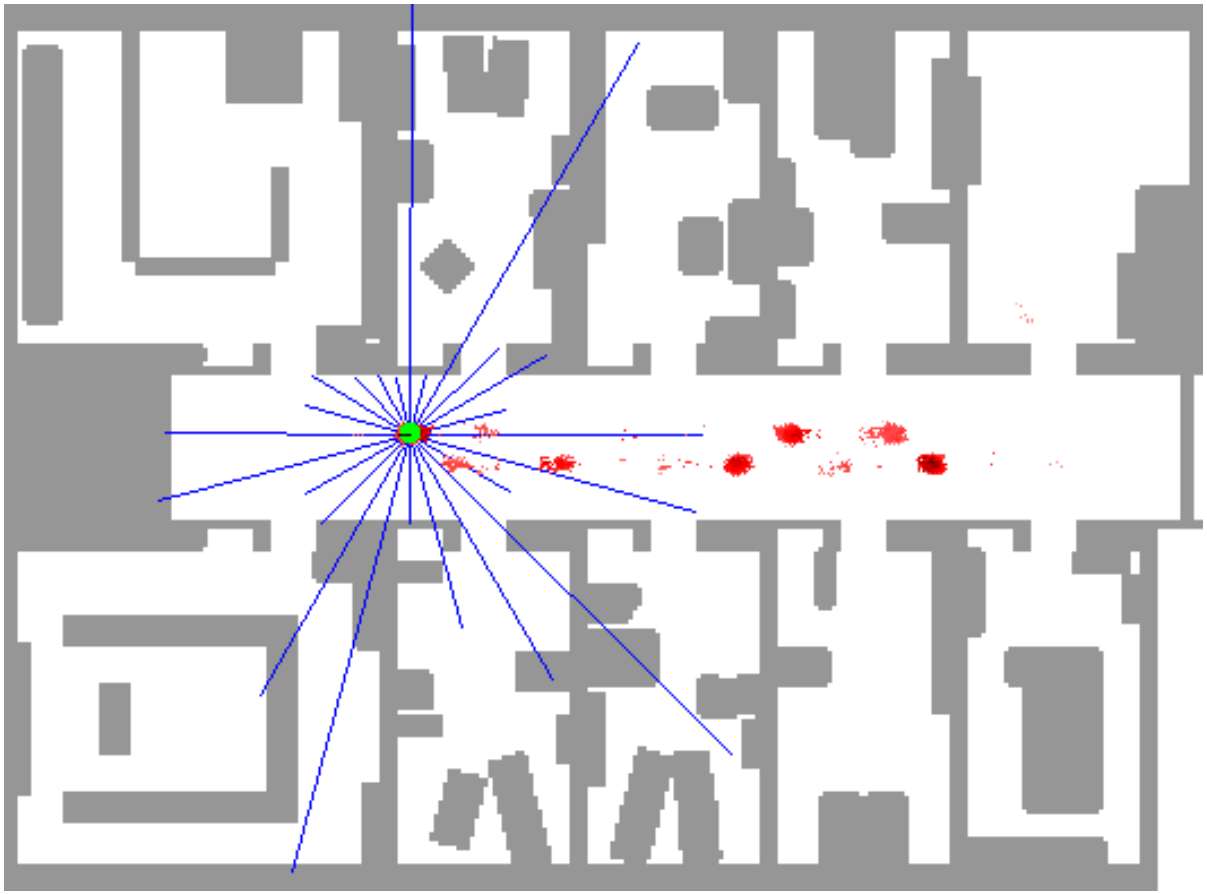




- The robot stores and updates a *probability distribution* representing its uncertain position estimate.

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:

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

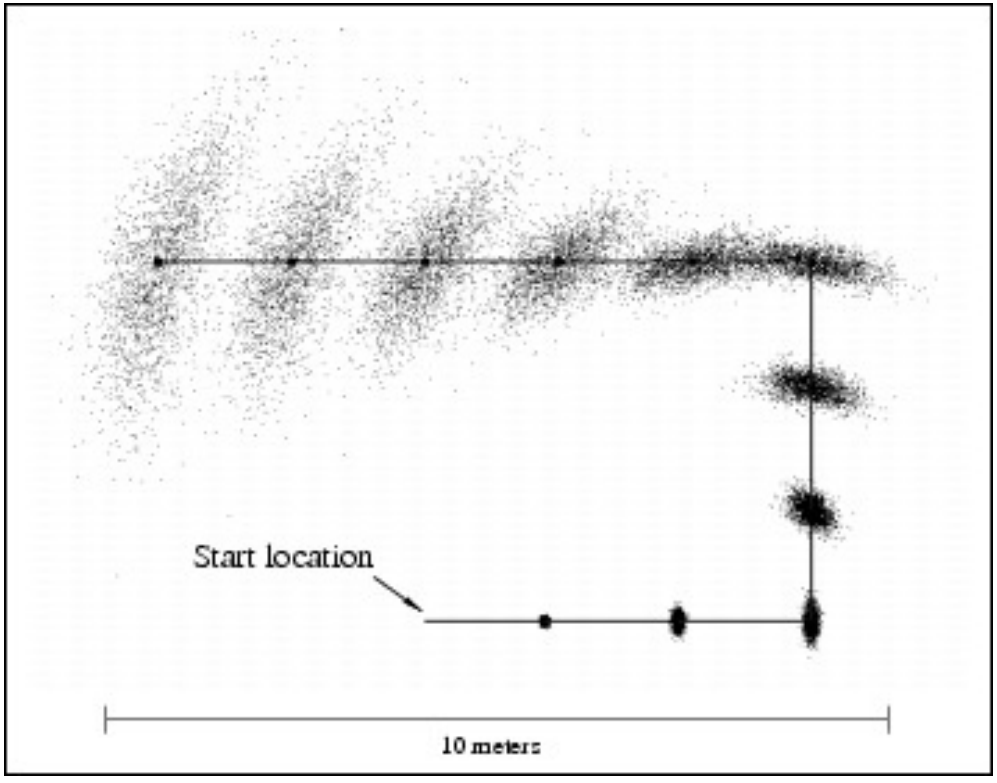
  for $i = 1$ to N. A typical value might be $N = 100$.

- 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.

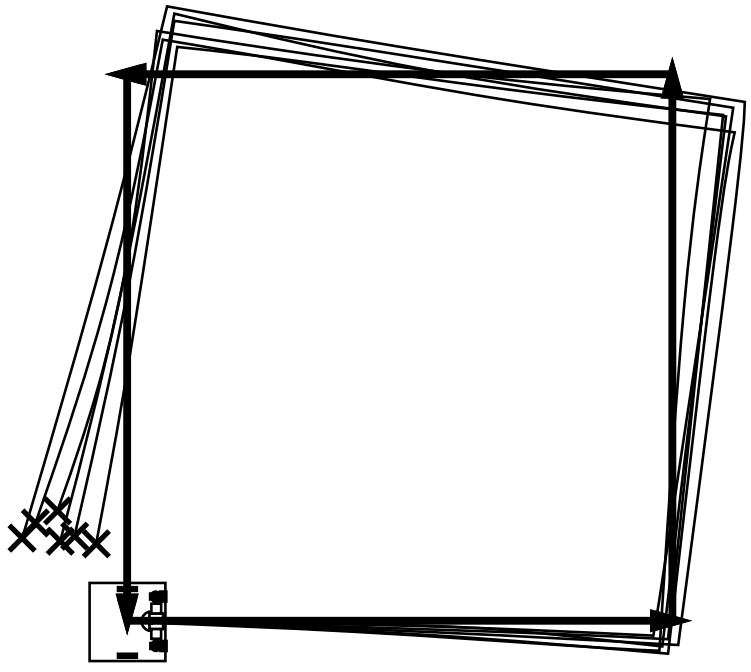


Steps in Particle Filtering

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 sample for each particle, which causes the particles to spread out.

Motion Prediction

- We need to find the appropriate standard deviations of the Gausssian distributions from which to sample e, f and g. Make some initial estimates; then adjust by looking at the particle spread over an extended motion and match the distribution to experiments (see practical sheet).
- If your robot is to move through variable step sizes, the values of these standard deviations should change with the distance move. The correct way to do this is to scale the **variance** proportional to the linear or angular distance moved — variance is additive, so for instance two 1m steps will cause the same spread as one 2m step.

Measurement Updates

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

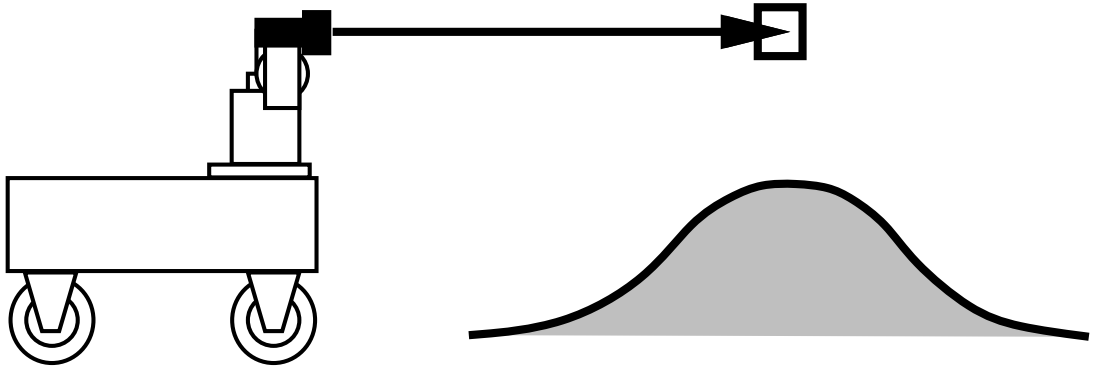
$$P(\mathbf{X}|\mathbf{Z}) = \frac{P(\mathbf{Z}|\mathbf{X})P(\mathbf{X})}{P(\mathbf{Z})}$$

- 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.

Inferring an Estimate and Position-Based Navigation

- At any point, our uncertain estimate of the location of the robot is represented by the whole particle set.
- If we need to, we can make a point estimate of the current position and orientation of the robot by taking the mean of all of the particles:

$$\bar{\mathbf{x}} = \sum_{i=1}^{N} w_i \mathbf{x}_i \ .$$

- i.e. the means of the x, y and θ components are all calculated individually and stored in $\bar{\mathbf{x}}$.
- The robot could use this for instance to plan A $\rightarrow$ B waypoint navigation.

This week's practical: Simple Sensor Control Loops and Wall Following

- Example wall follower: `https://www.youtube.com/watch?v=BU9k5Z0CKjs`. We will try to do even better with smooth proportional gain!