The Particle Filter

COSC450

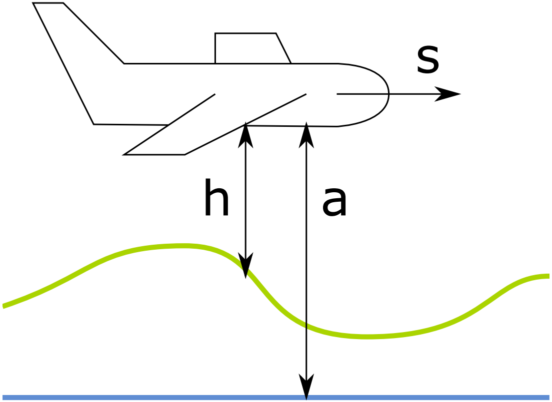
Particle Filter – Motivating Example

Suppose we are in an aircraft

- We have an elevation map
- We measure our speed, height above ground, and altitude above sea level
- Can we figure out where we are and where we are going?

Again we have:

- Measurement, $\mathbf{m} = \begin{bmatrix} s & h & a \end{bmatrix}^\mathsf{T}$
- State, $\mathbf{v} = \begin{bmatrix} x & y & z & \frac{dx}{dt} & \frac{dy}{dt} & \frac{dz}{dt} \end{bmatrix}^\mathsf{T}$



Particle Filter – Motivating Example

Next state is a function of current state

$$\mathbf{v}_{t+1} = \begin{bmatrix} x_t + \frac{dx}{dt} & y_t + \frac{dy}{dt} & z_t + \frac{dz}{dt} & \frac{dx}{dt} & \frac{dy}{dt} & \frac{dz}{dt} \end{bmatrix}^\mathsf{T}$$

The map gives a measurement function from the current state

$$\mathbf{m}_t = \begin{bmatrix} s_t & h_t & a_t \end{bmatrix}^\mathsf{T} = \begin{bmatrix} \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + \left(\frac{dz}{dt}\right)^2} & \mathrm{map}(x, y) & z \end{bmatrix}^\mathsf{T}$$

But we can't use the Kalman filter (without a good initial estimate of the state)

- Many places on the map have the same height
- This makes the distributions (highly)non-Gaussian – multi-modal distributions

The Particle Filter

Represent state by a set of particles

- Each is a hypothesis of the state
- They are samples from a distribution
- No assumptions about distribution

Particles have an associated weight

- A measure of their 'goodness'
- Ideally a probability they are correct
- Often a heuristic function

Informal algorithm

1. Initialise n particles, $\mathbf{p}_i$
2. Make a measurement, $\mathbf{m}$
3. Compute weights, w_i for each particle
4. Randomly pick particles based on w_i
5. Update their states (and add noise)
6. Goto 2

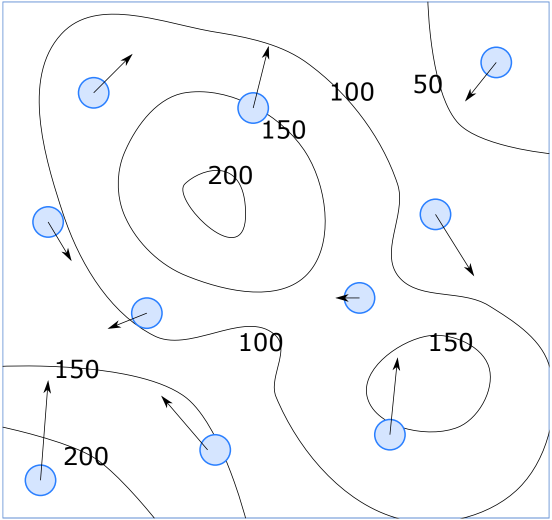
Particle Filter – Initialisation

Initialise the filter

- We have no prior knowledge
- Scatter particles through state space
- We'll use 10 for ease of visualisation
- Would probably need many more

Each particle

- Is a value for the state, **s**
- Represents a guess of the state
- Which ones are good guesses?

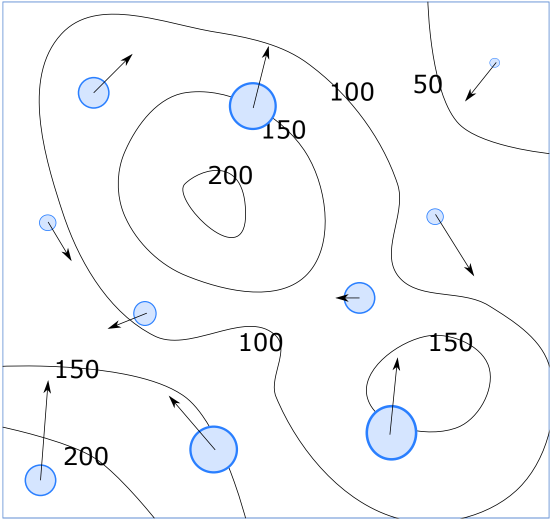


Particle Filter – Measurement

We now make a measurement:

$$\mathbf{m} = \begin{bmatrix} s \\ h \\ a \end{bmatrix} = \begin{bmatrix} 30 \\ 300 \\ 450 \end{bmatrix}$$

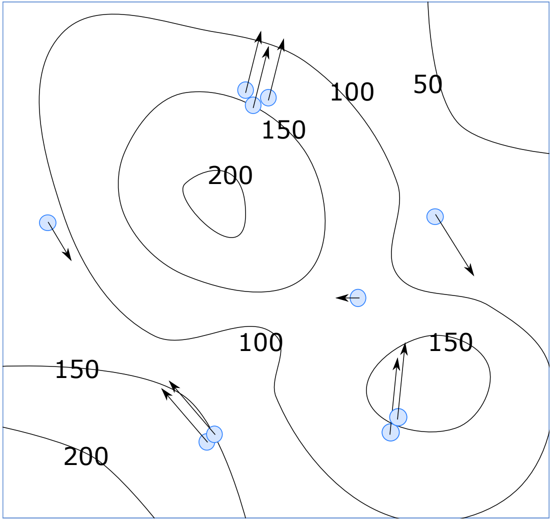
- This suggests the elevation is 150
- Some particles agree, others don't
- Use this to weight each particle



Particle Filter – Sampling

Now we resample by the weights

- We'll stick with 10 particles
- Each is a copy of an old one
- Particles can be duplicated
- High weight particles are *more likely* to be copied
- Low weight ones *might* be picked



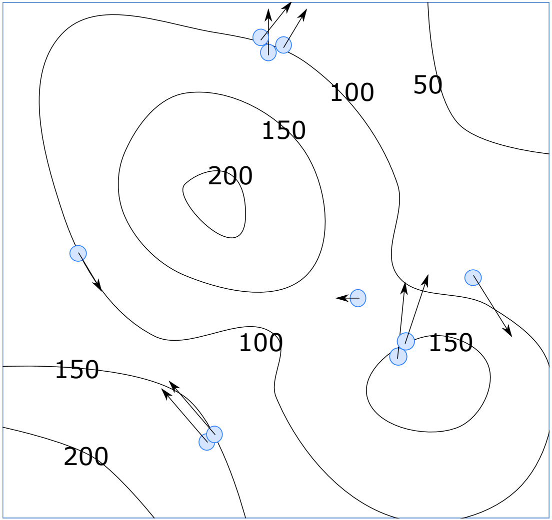
Particle Filter – Update

Next, update samples one time step

- Use the state update function
- Random noise added to states
- This spreads particles out to search different possibilities

We can then start to iterate

- Measure and compute weights
- Resample the particles
- Update and add noise



We can now give a more formal definition of the particle filter

- The mathematics showing this is valid is quite complex
- The implementation is quite simple – arrays of particles

The model for the particle filter looks much like the EKF:

$$\mathbf{s}_t = f(\mathbf{s}_{t-1}) + \mathbf{q}_t \qquad \mathbf{m}_t = g(\mathbf{s}_t) + \mathbf{r}_t$$

- f and g are functions, and need not be linear
- $\mathbf{q}$ and $\mathbf{r}$ are zero-mean random errors, but need not be Gaussian
- We do, however, need to have a model for $\mathbf{q}$ and $\mathbf{r}$

Particle Filter Algorithm

Step 1: Update

- We have n particles $\mathbf{s}_i$ at time $t-1$
- Predict the updated state as

$$\mathbf{s}_{i,t}^{-} = f(\mathbf{s}_{i,t-1}) + \mathbf{q}_{i,t}$$

- The value $\mathbf{q}_{i,t}$ is drawn at random from the known distribution of $\mathbf{q}$s

Step 2: Weight the particles

- We make a measurement, $\mathbf{m}_t$
- Predict measurements for each particle

$$\tilde{\mathbf{m}}_{i,t} = g(\mathbf{s}_{i,t}^{-})$$

- The difference $\mathbf{m}_t - \tilde{\mathbf{m}}_{i,t}$ has distribution of $\mathbf{r}$
- Use this to determine the probability (weight, w_i) that each particle is correct

Particle Filter Algorithm

Step 3: Resample

- Independently select particles n times
- *Not* the same as select n particles
- Probability of picking particle j is

$$p(\mathbf{s}_{i,t} = \mathbf{s}^{-}_{j,t}) = \frac{w_j}{\sum_{k=1}^{n} w_k}$$

This is *random* selection

- The 'best' particle might be missed
- Low weight particles might be sampled

A Monte Carlo approach

- It wins on average
- Large n approximates any distribution

There isn't a single estimate of $\mathbf{s}$

- All particles are estimates
- Can take average of all samples
- Can take highest weight sample
- Are these good things to do?