SLAM

SIMULTANEOUS LOCALISATION AND MAPPING

ROBOTICS (COMS4045A / COMS7049A)

What is SLAM?

- You've already looked at the localisation problem.
  - Need to keep track of your location given a map, control and observations
- SLAM is similar except you have no map
  - You only have control and observations and must build a map at the same time as you're keeping track of your location in it
- This is a lot harder in theory, but surprisingly similar in practice

Particle Filters

- You did localisation using particle filters, which allowed you to model an arbitrary distribution
- Each particle is a hypothesis. For localisation, the hypothesis is about your position
- We iteratively evaluate the plausibility of each hypothesis as new data comes in, and kill off bad hypotheses, and spawn new ones to deal with noise

Particle Filters

- The steps in particle filtering for localisation thus look as follows:
- Create n particles representing hypotheses of position and orientation
  - The iteratively:
    - Update the positions and orientations in the particles based on the motion model
    - Take an observation
    - Update the weight of the particle based on how well the position and orientation correlate with the observation
    - Resample n particles from the particles, biasing towards high weight particles, allowing for a particle to be selected multiple times

Particle Filters

- So what happens here?
- Particles are initially random
- Bad particles are killed, causing much tighter hypotheses
- Hypotheses disperse due to uncertainty in the motion model
- The powerful thing is that this is keeping track of temporal correlations, so as soon as a hypothesis leads to something unreasonable, the whole time series is killed!

SLAM

- Particle filtering for SLAM is similar
- We will look at SLAM using landmarks, which is a common way of doing it
- These landmarks could be things like corners or visual features in the observations as long as it’s easy to identify features regardless of orientation or scale
- For the purpose of this discussion, we’ll use corners as our landmarks

Particle filtering for SLAM

- The important thing is that we don’t have a map
- So each particle must now contain a hypothesis about the position and orientation of the robot (x,y,θ) as well as a map.
- The algorithm must then update the position on the basis of the motion model, and the map and particle weight on the basis of the observations

Particle Filters for SLAM

- The steps in particle filtering for SLAM thus look as follows:
- Create n particles representing hypotheses of position,orientation and map
  - The iteratively:
    - Update the positions and orientations in the particles based on the motion model
    - Take an observation
    - Update the map based on the landmarks in the observation
    - Update the weight of the particle based on how well the observation correlated with the map
    - Resample n particles from the particles, biasing towards high weight particles, allowing for a particle to be selected multiple times

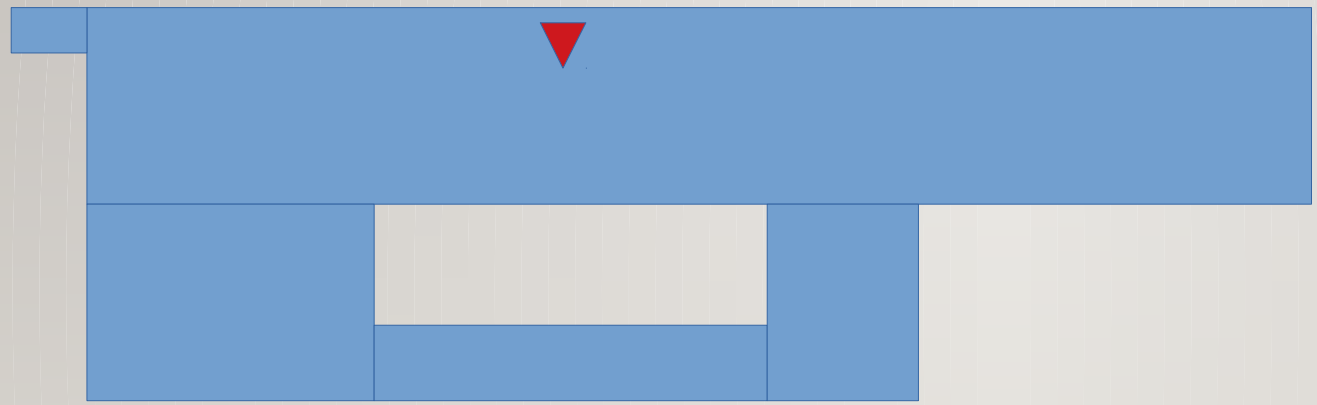
Create n particles…

- Creating the particles is easy. Since we don’t have a map, there’s no reason to pick any position over any other. So all particles have (x,y,θ) of (0,0,0)
- Since there is no map yet, they can all just have an empty map.
- So you have a huge pile of particles that are fundamentally equivalent

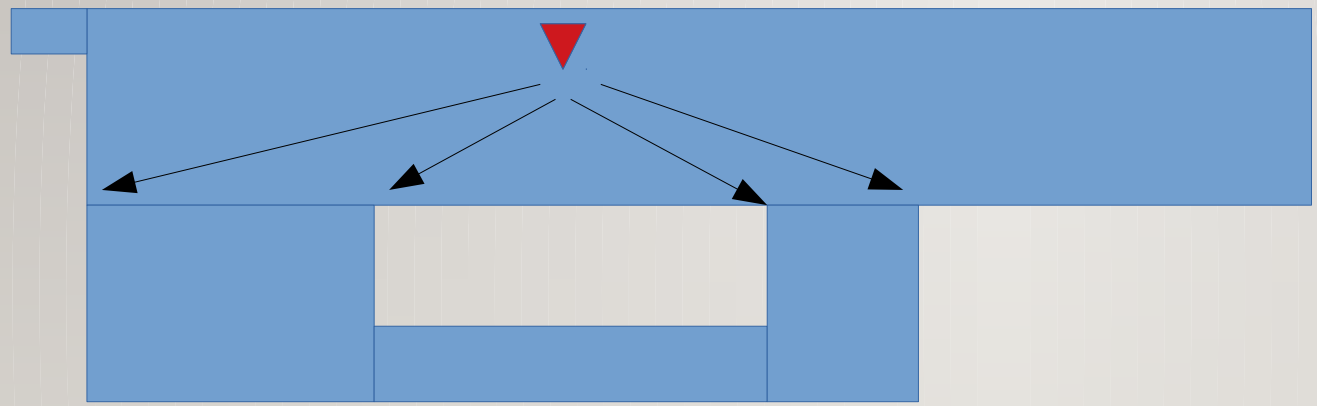
Iteration

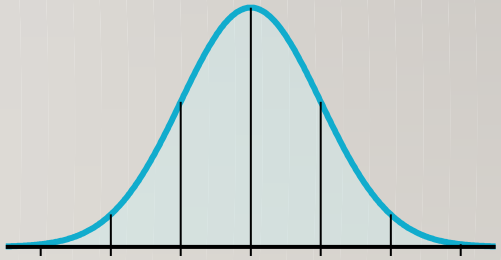
- How the particles diverge is because there is noise in the motion and observation models
- When we apply the motion model, we will draw a value from the distribution, which will be a different draw for each particle. We will thus end up with multiple (x,y,θ) hypotheses
- Now let’s look at what happens to the particles in each iteration

Example using corner landmarks



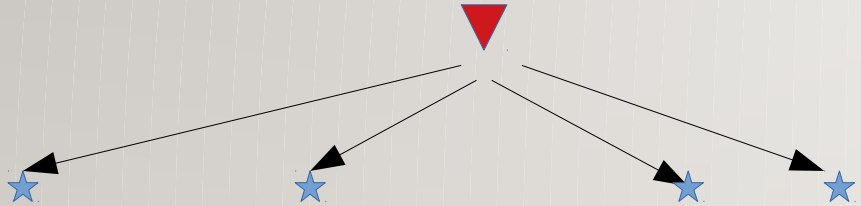
Example using corner landmarks

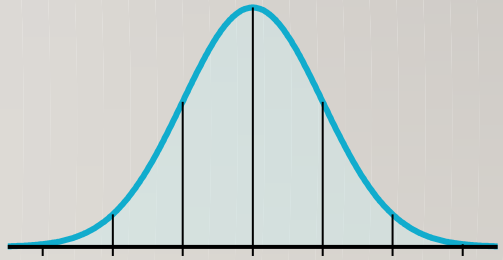




- We can see some corners in our observation
- Calculate the positions of the landmarks based on our position and orientation and our observation model
- Every landmark will have a mean and a variance, indicating uncertainty

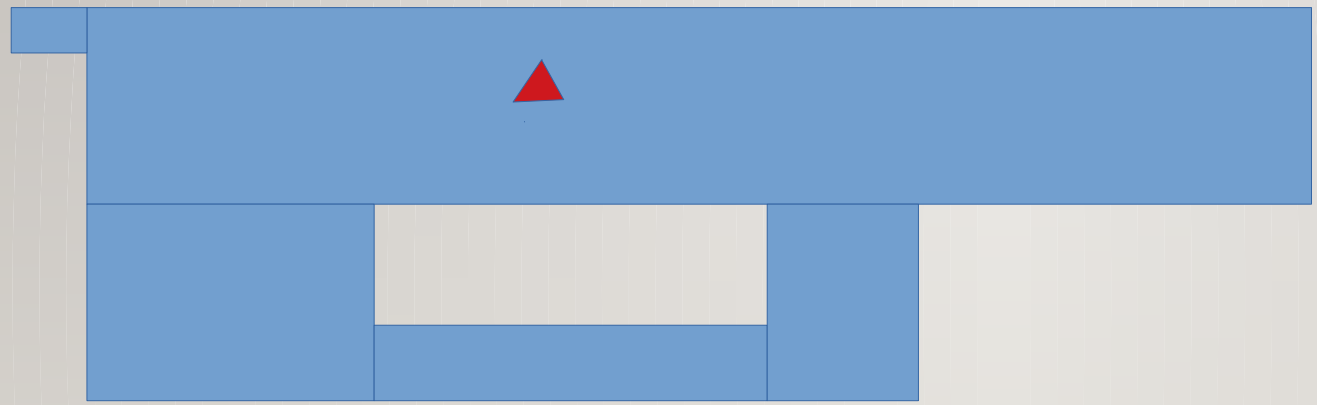
Example using corner landmarks





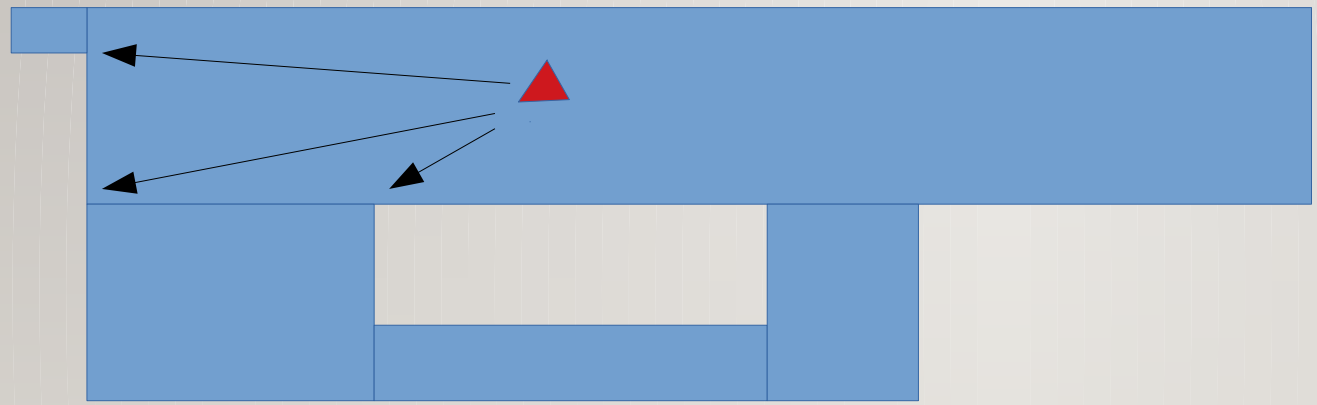
- Create a map, that just stores the positions of the landmarks, with their variances
- Note that our map only contains corners, so it really can’t see walls or anything. So all we have in the map are the four landmarks we observe.

Example using corner landmarks



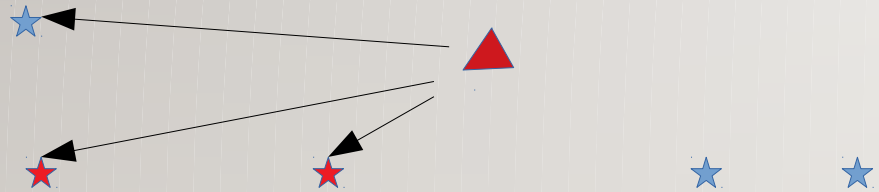
- Then we draw a sample from our motion model, given our control. This will move the particle along, and will cause particles to disperse due to noise

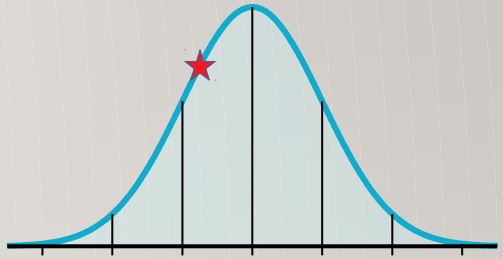
Example using corner landmarks



- Take an observation
- Note that we have to do a transformation to get the position of the landmarks as we have moved and rotated (Transformation matrices, trig)

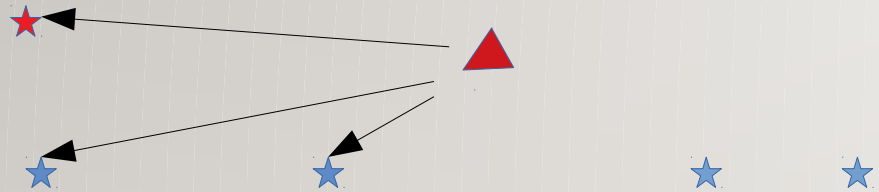
Example using corner landmarks

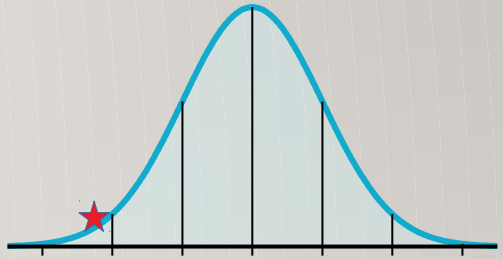




- How well do the landmarks correlate to the existing ones?
- Use the mean and variance of the landmark, and a threshold probability to indicate how likely it is that the landmark would be observed at the position we're seeing
- If we believe we're seeing a landmark we've seen before, update to reduce variance (See Kalman Filters)

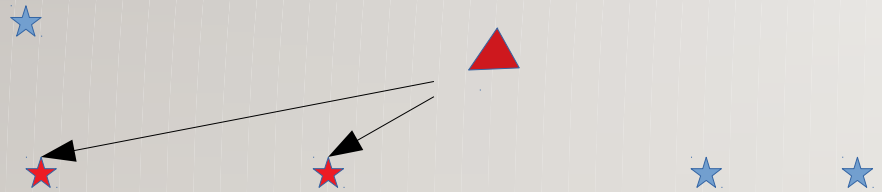
Example using corner landmarks





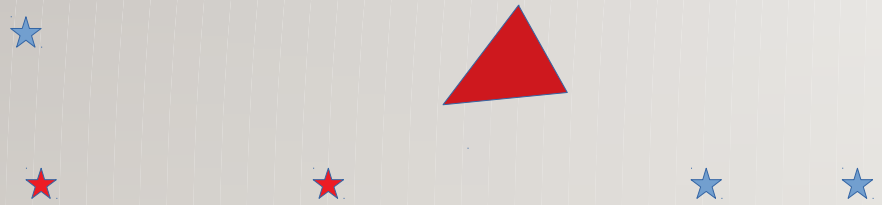
- If we’re seeing a new landmark, add it to the map

Example using corner landmarks



- Now to update the weights, we look at the landmarks that did correlate to existing landmarks
- We can pull a probability based on the mean and variance of the landmark, and this is how we decided it was an existing landmark
- Now, we multiply all the probabilities of the observed landmarks to get a probability of the particle, which will function as our weight

Example using corner landmarks



- Then we resample from the weighted set of particles, causing particles whose landmark observations did not correlate with the expectations from their own hypotheses to disappear. This one correlated very well, so it becomes gigantic
- In this way, we construct a map as well as keep track of our position simultaneously

Conclusion

- This lecture neglected the math and the specifics. The idea was to show you at a high level what SLAM is and how it works
- There are a number of different SLAM algorithms, so this was by no means comprehensive, just an example process of how to do SLAM