

Localisation

Noel Welsh

16 November 2010

Announcements

- No lectures next week, as I expect you will all be working towards the demonstrations
- Assignment 1 handed out at end of lecture.

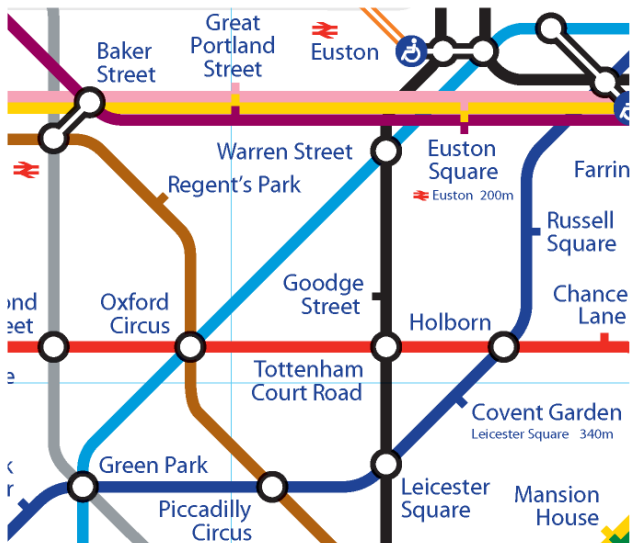
- Today we're mostly going to look at *localisation*
- Will also discuss the first assignment and the project writeup

How to Win Big

- The writeup is 4 times more important than the demonstration
- The writeup asks you to demonstrate you can:
 - 1 Choose appropriate methods for the key components of your robot (i.e. pay attention to lectures)
 - 2 Apply the scientific method to assess these components
 - 3 If performance is inadequate make sensible modifications and reassess, as time allows
 - 4 Present the above in a clear and coherent manner
- The process is more important than the end result
- Spend your time appropriately

- *Localisation* solves the problem of “where am I?”, *given a map*.
- What is a map?
- A continuum from pure *topological* maps to pure *metric* maps.

A Topological Map



Topological and Metric Map

- A *topological map* is one that is not metrically accurate.
- A *metric map* is one that precisely describes the location of all objects of interest.
- These definitions aren't really precise. Furthermore, there is a large space of *hybrid maps* that are an important area of research.

Dead Reckoning

- If we had odometry, we could localise.
- Is this going to work?

- Without reference to external landmarks, the error from dead reckoning will constantly grow.
- Dead reckoning cannot be used as the *sole* means of localisation.

Localisation in Topological Maps

- Below is a simple topological map. It consists of nine places which are connected in the obvious way. Dark lines indicate walls the robot can perceive. How could the robot localise in this map?

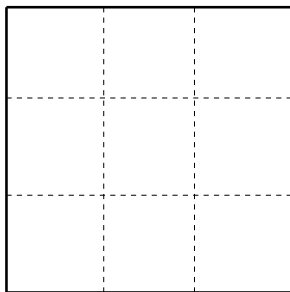


Figure: A simple topological map.

- There are two sources of information:
 - The observations the robot makes.
 - The controls we have sent to the robot.
- We should integrate these two sources.

Dealing with Observations

- Assume we can answer for each place $P(Obs|Place)$. That is: the probability of any observation we have seen, given we assume we are in a particular place.
- We can calculate this for all the places in the world.

- Assume we can answer for each place $P(\textit{Place}' | \textit{Control}, \textit{Place})$. That is: the probability of arriving in any particular place given the control sequence and the previous location of the robot.

Combining the Information

- $P(\text{Place}' | \text{Obs}, \text{Control}, \text{Place}) = P(\text{Obs} | \text{Place}') P(\text{Place}' | \text{Control}, \text{Place})$
- We can calculate this for all places. Normalise and we end up with a probability distribution over places. This is known as a *belief* or *information state*.
- This formulation is very similar to the MDP we saw last lecture.
- In fact, it is a POMDP.
- It is often difficult to translate from low-level control to the high-level movement given by a topological map. Thus we might drop the dependence on control, and assume, for example, the robot is equally likely to end up in any connected place. This is good enough for many cases.

Localisation in Metric Maps

- How do we localise in metric maps? Now the world is continuous, so the technique used for topological maps doesn't work so well.
- However, we can still apply the same general idea!

The Particle Filter

- Assume we have a motion model for our robot. E.g. assume turns and forward motion are corrupted by Gaussian noise. Can (and should) measure this from the real robot.
- Given the previous location, can *sample* from the motion model to approximate the distribution over the current location.
- Given each sample, can calculate the probability of observations as before.
- The samples are called *particles* and represent a hypothesis about the robots current location.
- This algorithm is called the *particle filter*. Also known as *sequential Monte Carlo* (SMC).

Particle Filter Example

Particle Filter example

Particle Filter Details

- Many algorithms. We'll look at *sequential importance resampling* (SIR), which is the basic algorithm.
- Need a *proposal distribution* Q , which given a control sequence, observation, and previous location gives a distribution over current location
 - Common choice is $Q(P'|O, C, P) = Q(P'|C, P)$ as it is easier to sample from.
- Each particle has an *importance weight* which is proportional to the probability we believe it is the true location

Particle Filter Algorithm

- Start with K particles (i.e. samples) with uniform weight.
- For each time step
 - Draw K new particles from the proposal distribution $Q(P'|C, P)$
 - For each particle, calculate the new importance weight
$$w_k = w_{k-1} P(\text{Obs}|P')$$
 - Normalise the weights to sum to one.
 - Compute the effective sample size $N_{\text{eff}} = \frac{1}{\sum_K w_k^2}$
 - If the effective sample size is less than some threshold, resample

- Replace the current set of particles with a new set constructed by sampling from the current set with probability proportional to weight.
- Set weights of new to particles to uniform $\frac{1}{K}$.
- Why?
- The problem of *weight degeneracy* occurs, when all but one of the weights becomes very low.
 - Waste computation on low probability particles
 - Don't adequately represent high probability hypotheses

Particle Filter Issues

- The accuracy of the particle filter is proportional to the number of particles.
- In 3D (x , y , and heading) the particle filter is ok.
- In higher dimensions need exponentially increasing number of particles.
- Alternative is to approximate the location with a Gaussian
 - Assume linear transition function
 - Assume hypothesised places always uni-modal
 - This algorithm is called the *Kalman filter*
 - Fast but not as powerful

- We have assumed a map is available.
- If we want to build a map and localisation at the same time, the problem is called *simultaneous localisation and mapping* (SLAM)
- There is a huge literature on the particle filter, and SLAM.