

Monte Carlo Localization

Susan Fox
COMP/NEUR 484
Fall 2004

`MONTECARLOLOCALIZATION` produces at each time step a set of “samples” which correspond to points in the robot’s map of the world. The samples are drawn from a probability distribution for the robot’s location that takes into account past samples, the uncertain results of the robot’s movement, the uncertain data from the robot’s sensors.

The samples are kept as “static” variables of the algorithm. That means, they belong to the algorithm, but maintain their values from one call to the next. Before the first call to the algorithm, the collection of samples is initialized by sampling the prior probability distribution for the robot’s location. In the worst case, the robot has no idea where it is, so every location in the space has equal probability and the initial samples are randomly scattered throughout the space.

There are three phases to each run through the Monte Carlo algorithm.

1. In the first phase, each of the old samples is updated to reflect the results of any movement the robot may have made. A new sample is drawn from the probability distribution $\mathbf{P}(\mathbf{X}_1 \mid \mathbf{X}_0 = S[i], A_0 = a)$, which defines a normal distribution in the map space for how likely the robot is to be at any particular location in the world, given that it started at the sample’s location and made a movement described by a . (*This is the motion model for the robot.*) Even in the case where the robot is staying still, there is still some probability that the robot is actually not quite at $S[i]$, so the new sample may not be identical to the old one.
2. In the second phase, each new sample is weighted to indicate how likely it is that the robot would read the sensor data it got if it were at the new sample’s location. For each separate piece of sensor data, the algorithm calculates the probability of reading that value, given the robot’s location as $S[i]$ and the sensor model for the robot. The overall weight is the product of these probabilities over all pieces of sensor data.
3. In the third phase, which is reduced to a single line in the pseudocode, the set of new samples is resampled based on the weight values. This is done with a technique often called weighted roulette wheel selection. On a normal roulette wheel, every number has an equal sized wedge of the wheel, so every number is equally likely. Here, each sample in the collection is assigned a wedge of the wheel proportional to its weight value. Then a new set of samples is chosen by spinning this weighted wheel N times and putting a copy of the selected sample into the new set. Notice that the same sample may be chosen repeatedly.

Also, notice that there is an element of randomness here: sampling from a probability distribution involves randomness guided by the distribution, and the weighted selection similarly involves randomness guided by the weight values. Randomness guided by probabilities is the hallmark of “Monte Carlo” methods of various sorts.

Here is the pseudocode from the book, heavily commented by me. The algorithm returns the new sample, as well as keeping it in variable S until the next round. Its inputs are a , the movement it made last time step; \mathbf{z} , a collection of M sensor readings ($z_1 \cdots z_M$); N , the number of samples being kept; $model$, the set of models needed, including a 2D+ model of the robot's prior position $\mathbf{P}(\mathbf{X} + \mathbf{0})$, a 2D+ model of the robot's motion $\mathbf{P}(\mathbf{X}_1 | \mathbf{X}_0 = S[i], A_0 = a)$, and a sensor model $P(Z | \hat{Z})$; and map , a map of the world.

The algorithm uses a local array W of size N for holding the weights it calculates, plus the static variable array S of samples.

```

MONTECARLOLOCALIZATION ( $a, \mathbf{z}, N, model, map$ )
    // Create  $N$  new samples based on robot's motion
1.  for  $i = 1$  to  $N$  do

        // Take motion into account and generate a probability distribution centered on
        // the most likely new location for the robot
        // Then sample that distribution, getting a new sample by generating
        // a random value guided by the distribution
2.  set  $S[i] = \text{SAMPLEFROM}(\mathbf{P}(\mathbf{X}_1 | \mathbf{X}_0 = S[i], A_0 = a))$ 

        // Calculate the weight values to reflect how likely the sensor data is,
        // given the new sample location
3.  set  $W[i] = 1$ 
4.  for  $j = 1$  to  $M$  do

            // Assuming robot is at  $S[i]$ , compute the distance the  $j$ th sensor ought to detect
5.  set  $\hat{z} = \text{EXACTRANGE}(j, S[i], map)$ 

            // Compute the probability of reading the sensor data  $z_j$ , if the actual distance is  $\hat{z}$ 
5.  set  $W[i] = W[i] \cdot P(Z = z_j | \hat{Z} = \hat{z})$ 

        // Use weighted roulette wheel selection to resample the old  $S$  values
6.  set  $S = \text{WEIGHTEDSAMPLEWITHREPLACEMENT}(N, S, W)$ 
7.  return  $S$ 

```