**Particle Filters**

When trying to find out what state we are in, we learned about Kalman Filters, which work for continuous state spaces. There are also Histogram Filters which work for Discrete State Spaces.

The Histogram Filters belief state could be multimodal (meaning that the distribution could have more than one peak), and the Kalman Filter is only Unimodal (only one peak in the distribution is possible. This is from linearizing equations and using the Jacobean, or the unscented method which uses points in the distribution to represent the whole thing). Kalman filters belief was a single Gaussian distribution.

In terms of **Efficiency**, meaning how well it scales with the number of dimensions of the state space and the number of storage it needs to assign, The **Histogram Filters method is exponential** (meaning it works well for lower dimension localization problems) , while **Kalman Filters are quadratic**.

All the storage required for the Kalman Filter was a vector for the mean, and the covariance matrix. All the computation in the algorithm is quadratic.

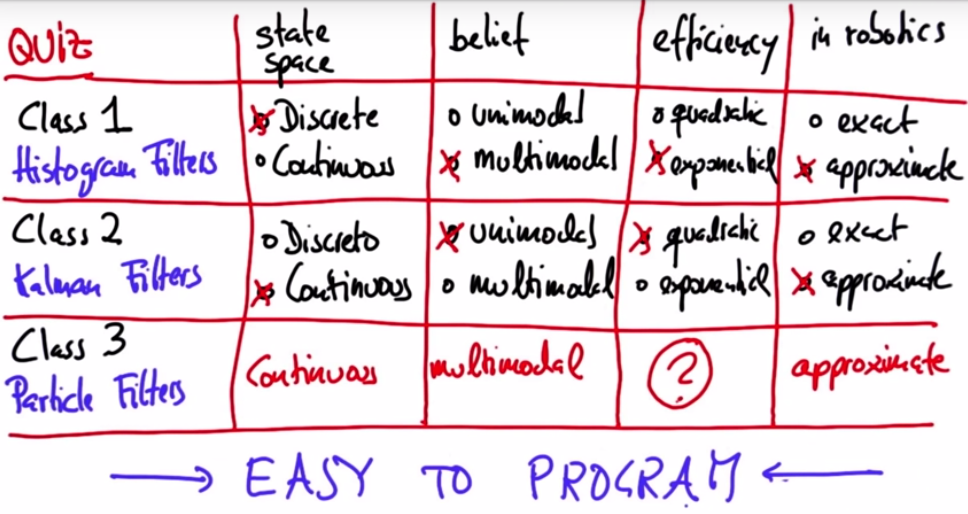
**In robotics**, Histogram Filters are considered approximate due to the fact that it assumes the world is discrete, so for localization, it will not result in an exact answer.

Kalman Filters are also approximate, due to the fact that it requires linearization. The error caused by linearizing the problem makes it an approximate answer. If the problem in question was a linear system, the Kalman Filter would provide an exact answer. In the real world, where there are Non-linear systems, the Kalman Filter is approximate.

But now we will learn about Particle Filters, which is yet another algorithm that can help us find out what state we are in.

A Particle Filter can represent a Continuous State Space, its Belief States are Multimodal, they give an approximate answer, but in terms of efficiency it is still questionable based on the application.

Particle Filters in some incarnations scale exponentially, so it is a mistake to represent particle filters with anything more than 4 dimension, but in tracking domains, they tend to scale much better so it is questionable. **The key advantage of Particle Filters is that it is Easy to Program**.



Particle Filter we will now create will maintain 1000 random guesses (Particles) of where in the world the robot is.

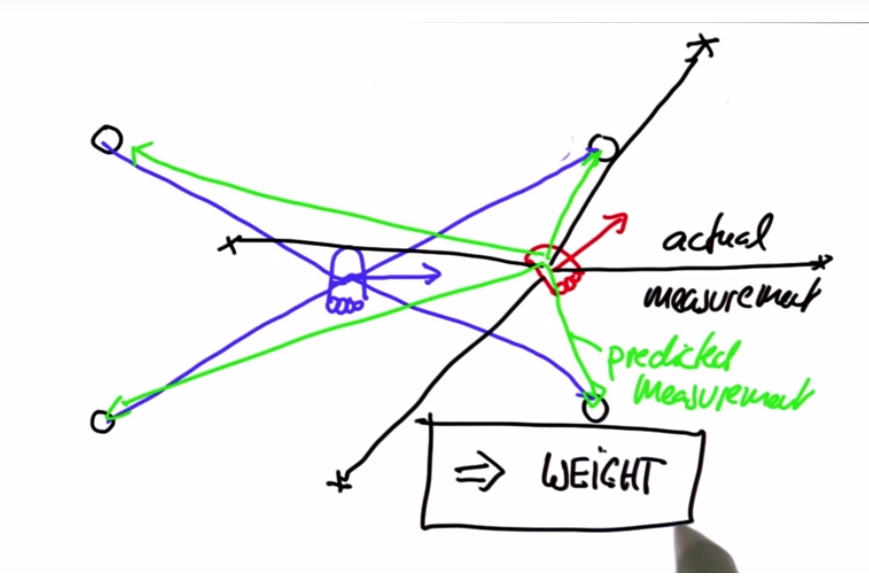
Each particle is the position (x,y), and the orientation(angle with respect to x axis).

Each particle is initialized with randomly with values within the space. So if the world is 100 by 100, the random guess is within 0-100 for x and y values.

Each particle has a measurement of how far it is from landmarks in the world.

Now let’s say our robot takes a measurement of where each of the landmarks is relative its current actual position. The particles that have similar measurement values are much more likely, and thus should have a higher **Importance Weight**.

The mismatch of the actual measurement to the landmarks and the predicted measurement is what makes this weight. The larger the weight, the more important that specific measurement is.

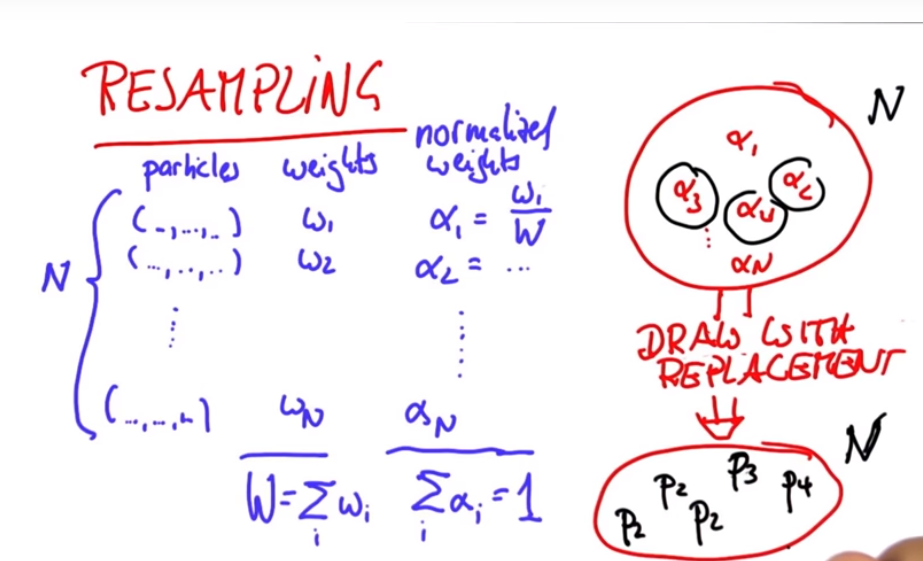


Each particle has its own weight. The probability that the particle survives is proportional to its weight.

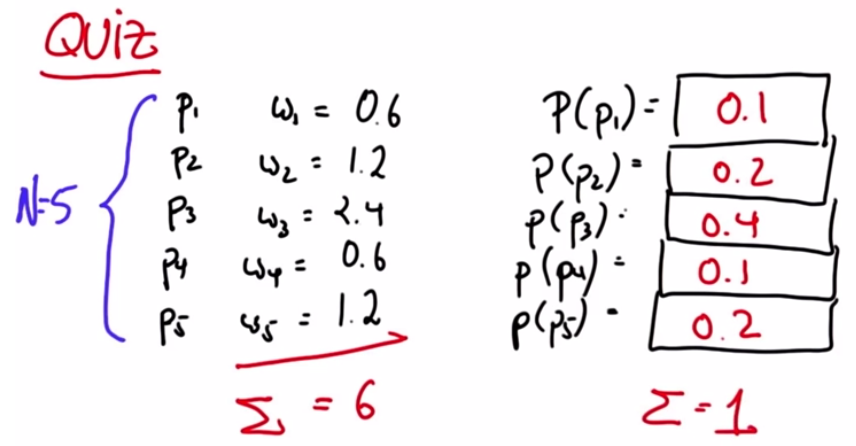
This means that after the process of **Resampling** which is randomly drawing and replacing N new particles from the old ones, in proportion to their importance weight.

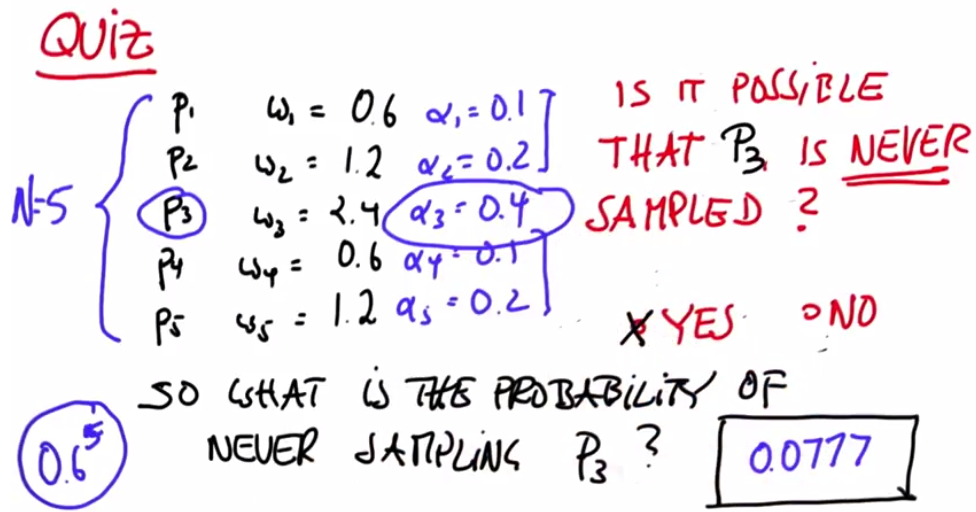
This means particles cluster around the actual robot measurements in the world.

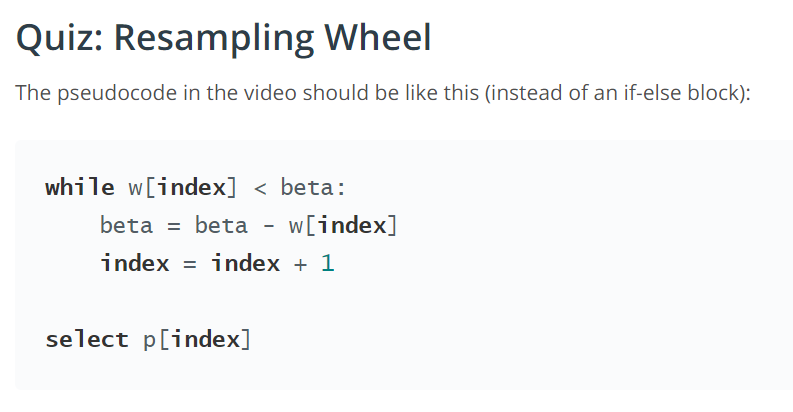
**Resampling**

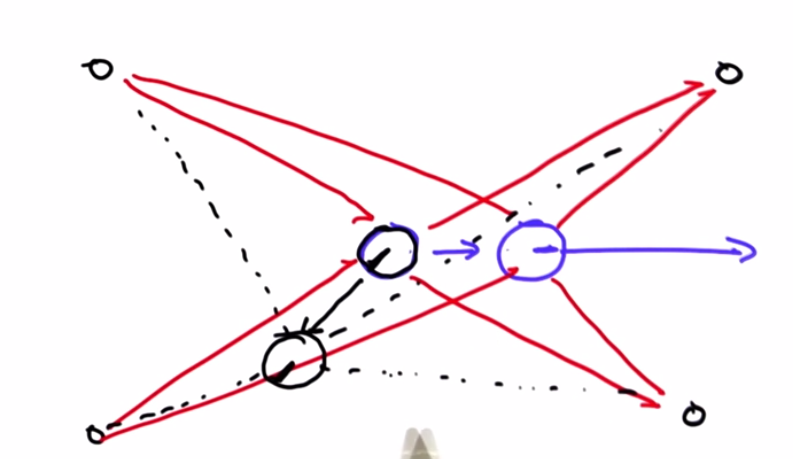


The frequency a particle is resampled is proportional to its normalized weight.





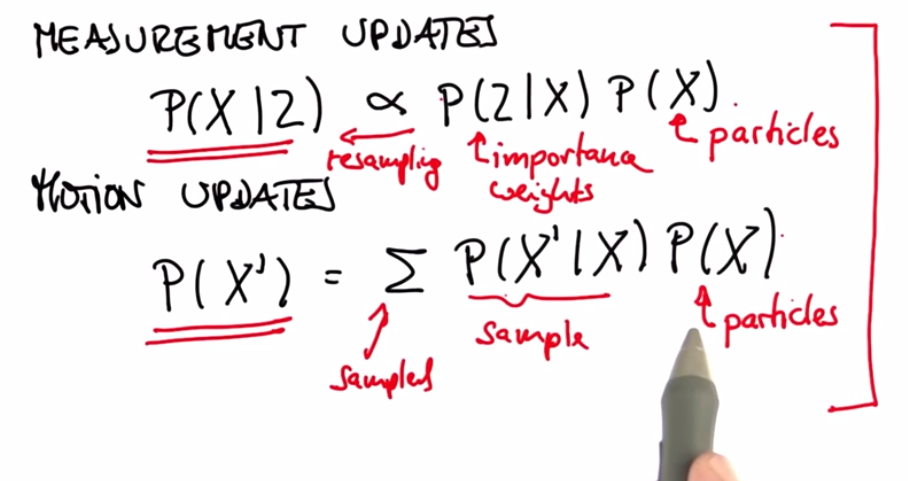




Orientation matters when we want to run the particle filter multiple times. If the initial orientation is different, the measurements are very different for each particle.

This results in them being resampled less, so orientation is handled inherently when resampling is done multiple times.

**Overall**



**Differences**

From the Google Car.

Robot model used in self-driving car is a Bicycle model. Sensor data is different. Instead of using landmarks, we use an elaborate road map, and then we take a single snapshot, and try to match it to the map, giving it a higher score.

