**Markov Localization**

Bayes Localization Filter (Markov).

A Bayes Filter.

Let’s think about how a car is localized.

We have a map of landmarks in a global co-ordinate system, the observations of landmarks in the cars local co-ordinate system, and information about how the car moves between 2 time steps.

These 3 things are known.

Observations: A vector of observations from time step 1 to t

Controls: A vector u that includes all control elements from time step 1 to t

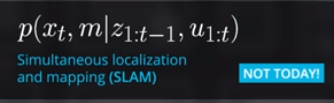
Map: A grid map of the global environment or a database with global feature points and lane geometry.

The map is time invariant. We want to know/estimate is the transformation between the local co-ordinate system of the car, and the global co-ordinate system of the Map. Which can then lead us to finding the position and orientation of the car in the world. We want an accurate belief of the state xt.



We want to estimate the probability distribution of xt, with the condition that we know all previous observations and controls. We also need to know the map, and understand that it is fixed and time invariant.

If we don’t assume the map is given and also want to estimate it, then the problem is called **SLAM**, **Simultaneous Localization and Mapping**.



**Now how does this data look?**

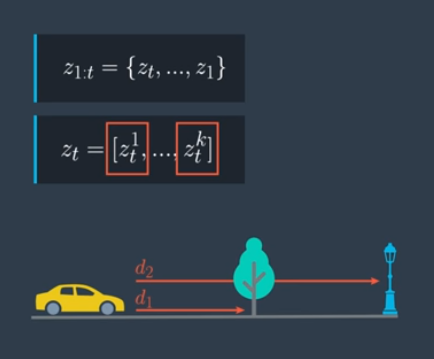
The **map**, m, for 1D Markov Localization, is the position of streetlamps, and trees in 1D. This is a landmark based map which is in general, sparser than a grid based map.

It is a vector of the position of these landmarks.



The **Observation** is the nearest k, scene static objects in the driving direction. The car can detect the distance to the nearest cars and trees in the driving direction.

The resulting observation list, which for each time step has a vector of size k, of the distance to the k nearest landmarks (trees and streetlamps).



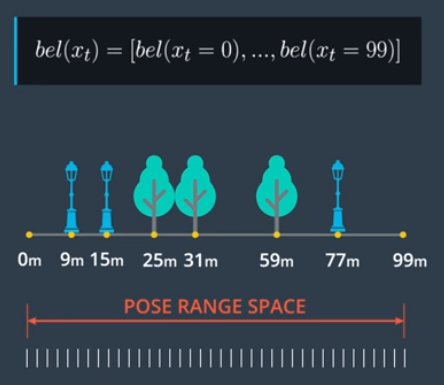
The **control vector** is the direct movement of the car between two consecutive time steps.

For the 1D case the control vector is the distance the car moved from the previous time step to the current time step.



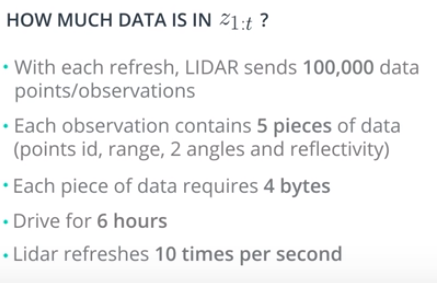
The **position/pose** of the car is somewhere in the mapped area, and since the map is discrete, the position could be anywhere in the mapped area. So if the map is a vector from 0 meters to 99 meters, the pose of the car is any integer between 0 and 99 meters.

That means that the belief of xt, is a vector of 100 elements that has the value of the probability at the corresponding position.

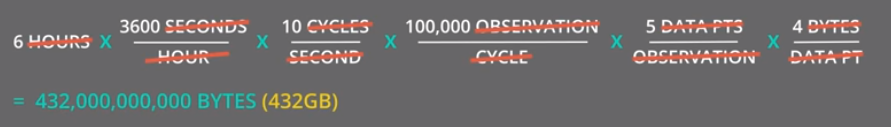


How much data do we require to keep as an observation?

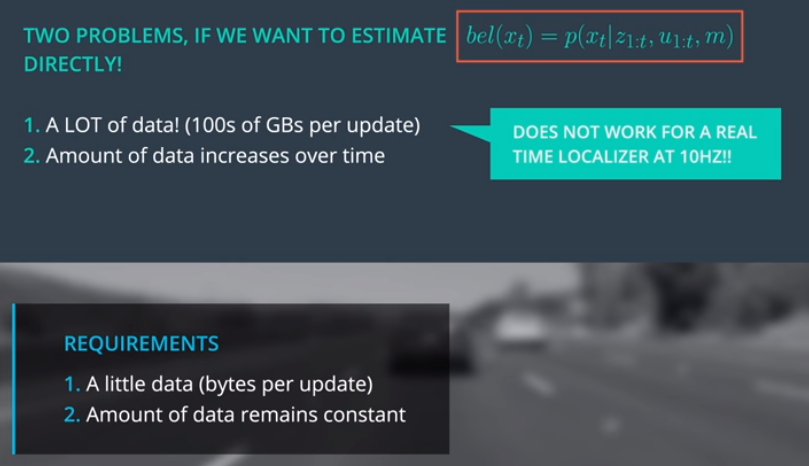
Let’s consider the trade-off here.



Answer: 100,000 \* 5 \* 4 \* 6\* 60 \* 60\* 10 = 432 GB of data.



There are two issues we need to solve in order to estimate the belief of xt.



What if we could just update our estimate at every time step so that we don’t need to keep data for the entire history of the system?

We can use Bayes rule to make an estimation that we can update.

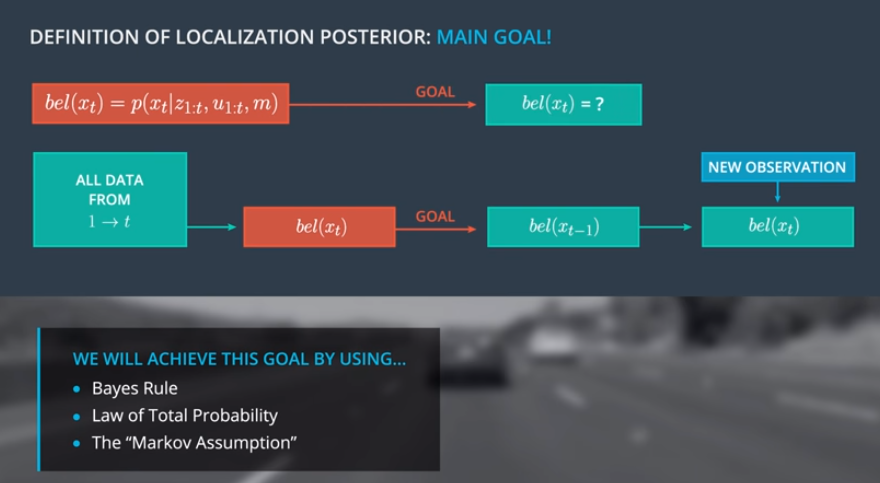
We don’t want all data from time 1 to t. We want to have a recursive state estimator.

Our goal is to change our state estimator to account for all previous data separately, and then update it with data from the current time step.

This type of system is called a Bayes Localization Filter or Markov Localization. It can allow us to not keep all previous observation and measurement data.

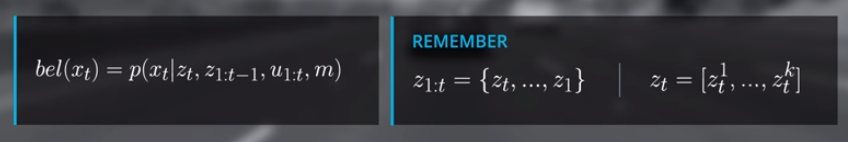
We need to use both Bayes rule and the law of total probability.

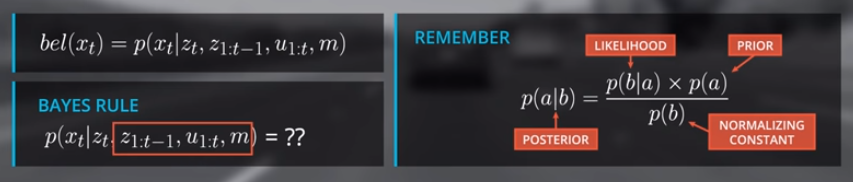
There is also something called The “Markov Assumption”, which involves making meaningful assumptions about the dependencies between uncertain values.

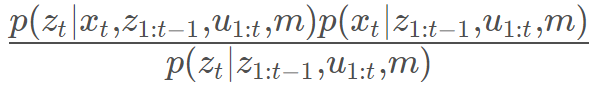


The goal is to define the posterior probability in a recursive way.

The first step is to split the estimator into previous observations and current observations.

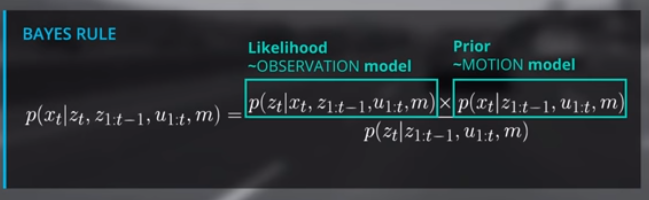






The result comes from applying Bayes rule, which involves swapping the state and observation at time t.

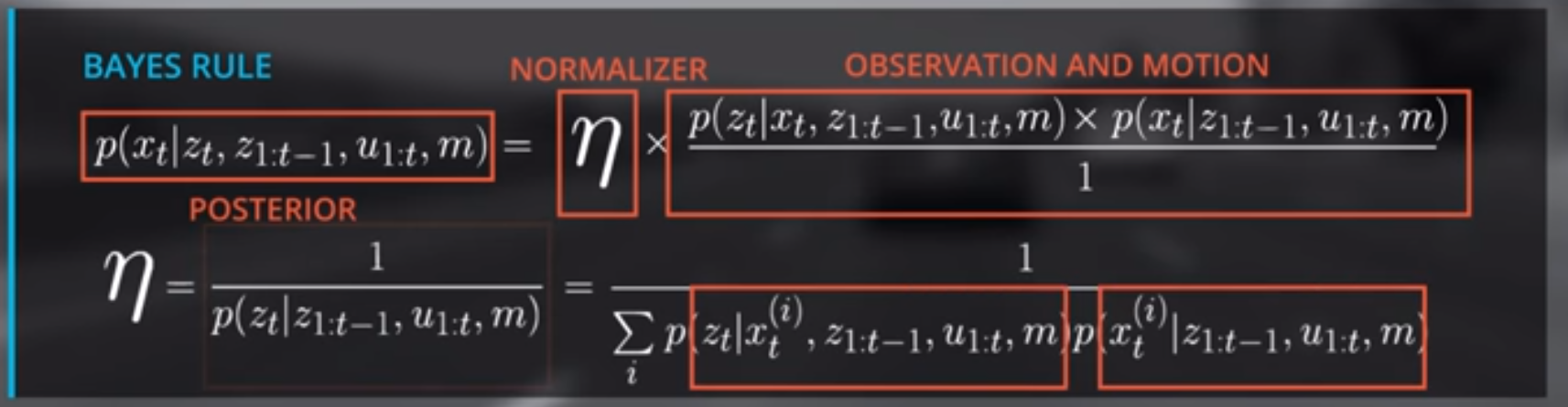
All other conditions are kept the same. It is fine to keep all the other conditionals without changing them.



The likelihood term is called the **Observation Model.** This describes the probability distribution of the Observation vector that the state, and all previous data along with the map is given,

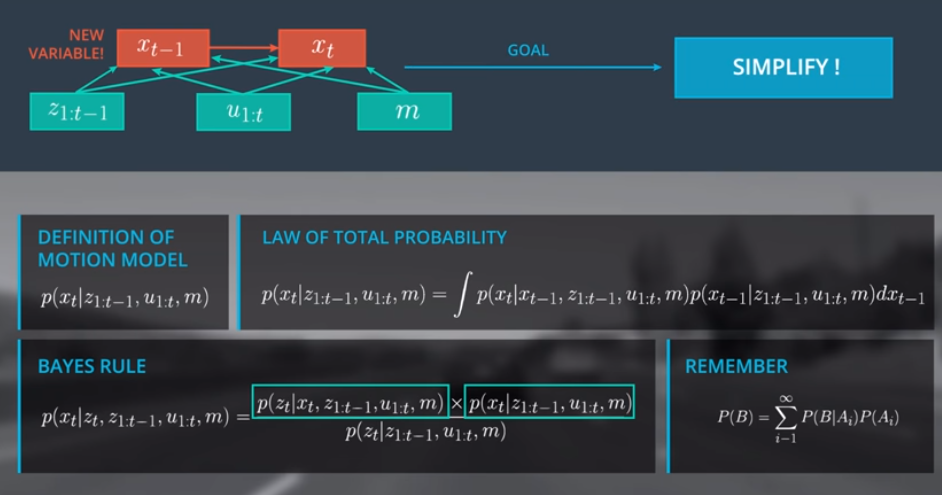
The Prior is considered the **Motion Model**, which is the probability distribution of the position given all other observations and controls are known. Keep in mind that no current observation is included in the motion model (Zt).

To simplify the normalization part, we define the normalizer as Eita.



The normalizer is composed of both the motion model and the observation model **(the sum of their product over all possible states)**, making them the only thing required to estimate beliefs.

Now when looking at the motion model itself, we see that we have no information about where the car was before or X(t-1). We can use the **law of total probability** given the value at the current time step, to find the probability for all previous time steps.



Integration over the state space of x(t-1). All states in the next time step estimated with the values of all previous time steps.

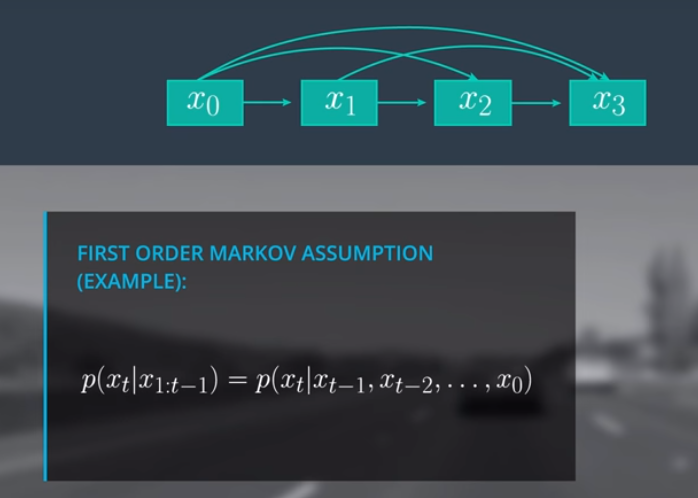
We can simplify the relationship between x(t-1) and x(t) by making assumptions.

**Markov Assumptions**



Both these assumptions are meaningful assumptions.

**First Order Markov Assumption**



First Order Markov Assumption is the assumption that the state x3 is best estimated using x2, and that x1 and x0 don’t give any useful information.

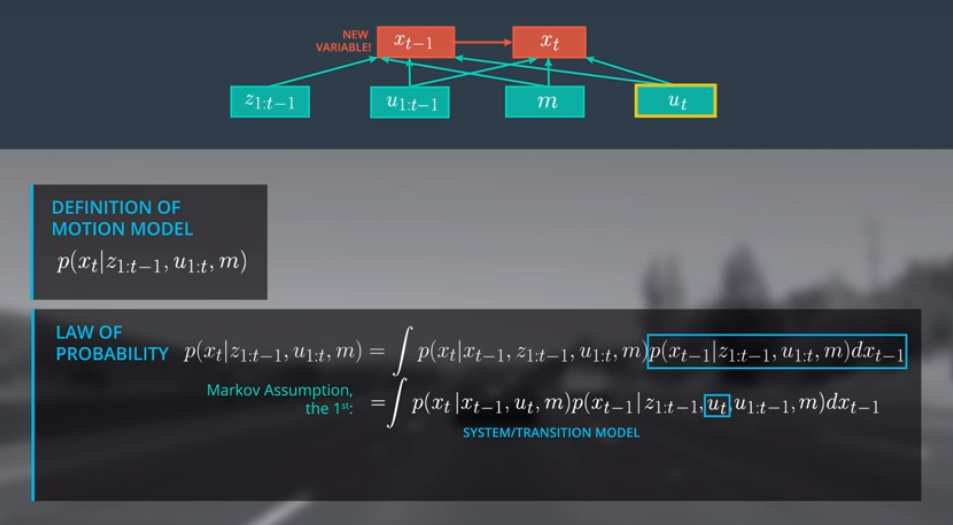
This assumption lets us change the posterior to this.



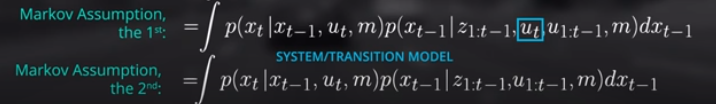


This assumption means that the current state only really needs to be estimated using the previous time steps state, and not the entire history. This also means we need a good initial condition.

This moves the state from the previous state x(t-1) to the current xt, which is why it is called the **System/Transition Model**.



**The other Markov Assumption used is that ut happens in the future so it has no effect on x(t-1).**



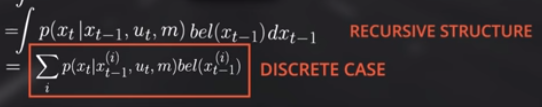
This separates the transition model into previous time step values, and current time step values.

This means that we have a recursive structure, which is exactly what we wanted so that we don’t need to keep track of every previous observation and measurement, we only need to keep track of the motion model, and the observation models probability distributions.

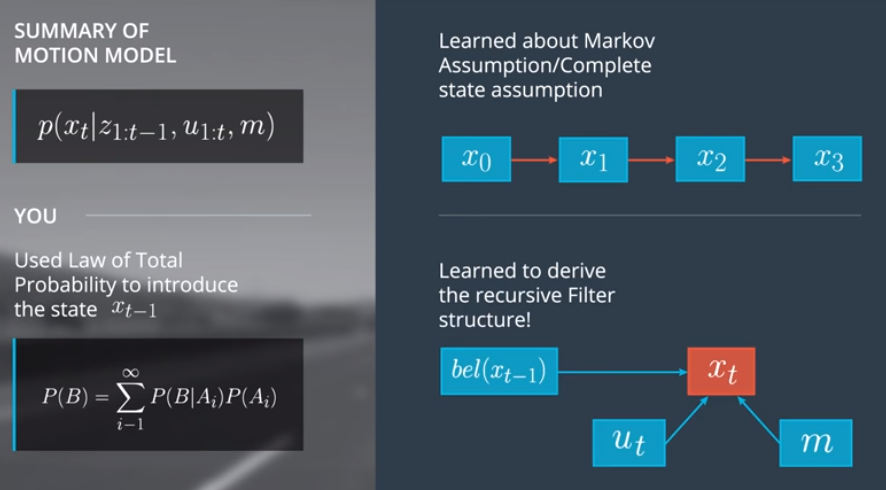


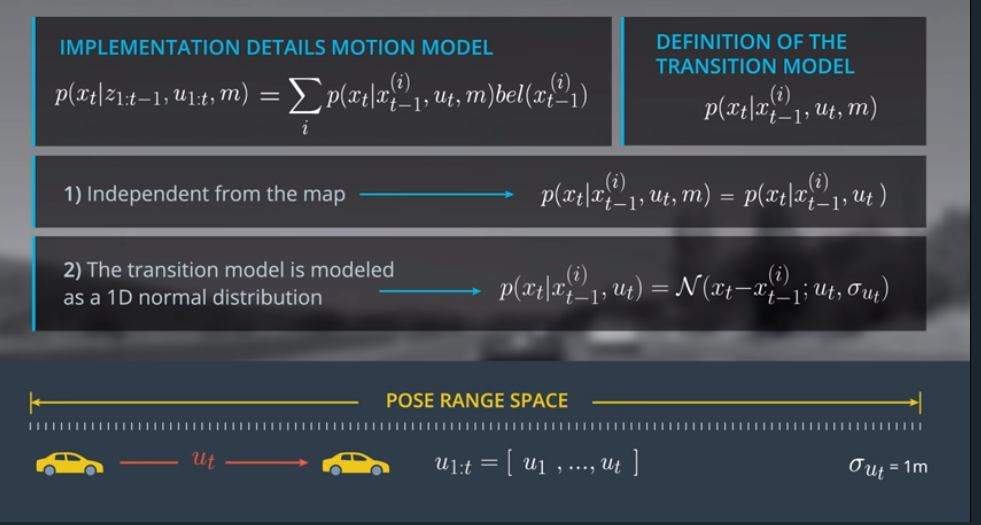
Because we have a discrete localization scenario we can get rid of the integral and make it a summation.

This is technically a convolution.



How the filter is initialized is very important. Some systems use GPS data to first initialize the distribution.

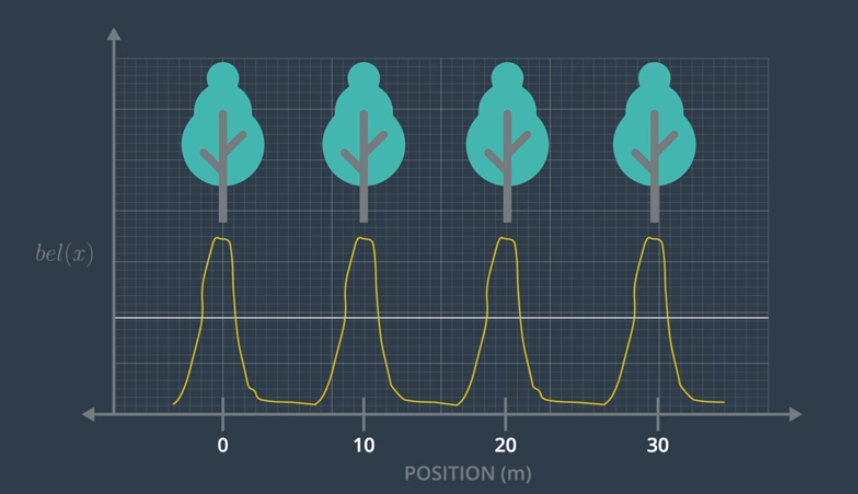


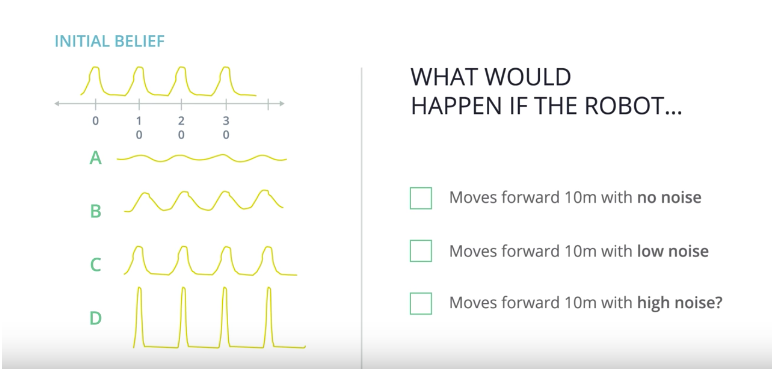


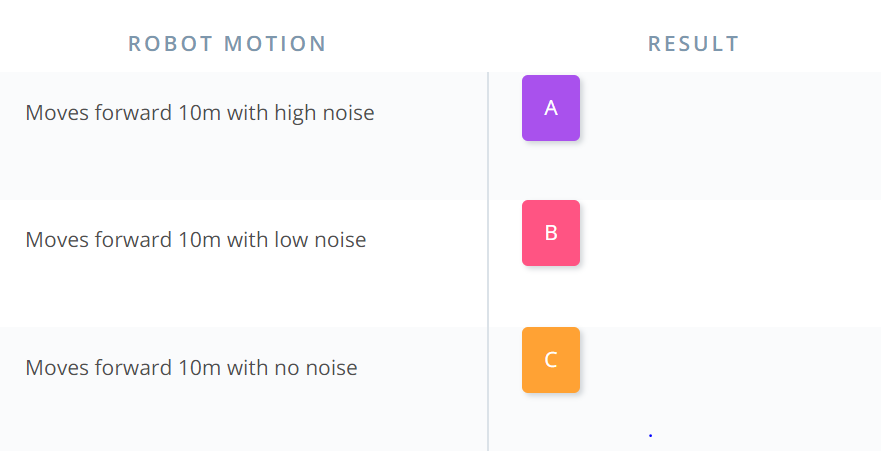
**Noise in Motion Model.**

At the beginning the car has no clue where it is, so the initial belief is a uniform probability density. Which is maximum confusion.

But what if we are close to a tree? Then the initial belief is this.



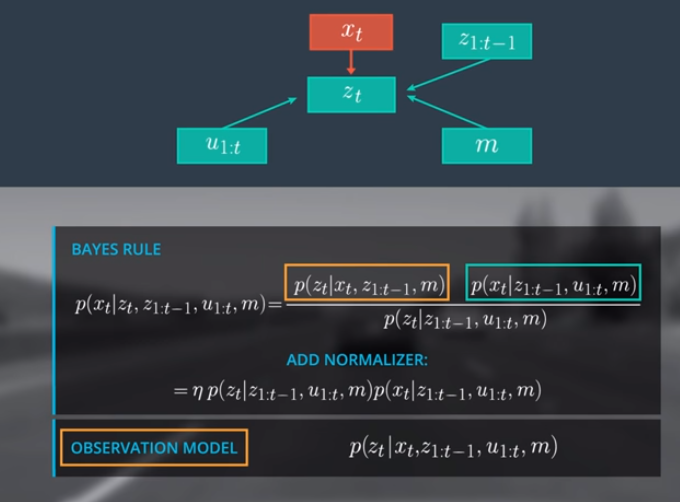




**Observation Model Introduction**

Now we can go back and try to implement the Observation Model.

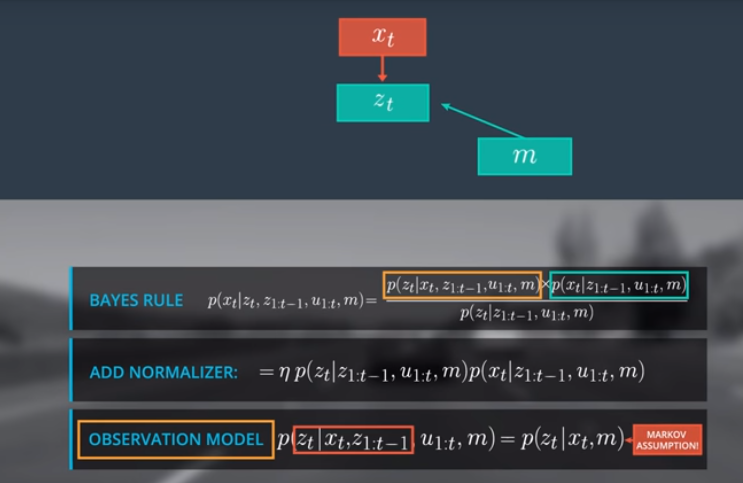
It is the probability distribution of the observation zt, given the current state, map, and previous observation and measurement values.



We used Markov Assumption to remove all past data to make our method recursive.

It was the assumption that the only relevant data used to estimate the state at the current time step is the state of the previous time step and not the entire history of states. And because the previous state takes into account previous measurements and observations, they are not required.

We can do the same thing with the Observation model.



The observation model is then greatly simplified.



Now looking at our observations, we need to remember that each time step is a vector of the current observations.

**If we assume that each observation is independent**, we can think of the observation model as a product of the individual probability distributions of each single measurement.

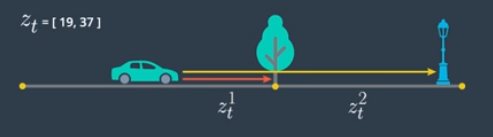
So now we need to thing about the probability distribution for each range measurement.

In general, each there are a lot of different observation models based on the sensor (RADAR, LIDAR, CAMERA, ULTRASOUND, etc).

Each sensor has its own specific noise behaviour and performance.

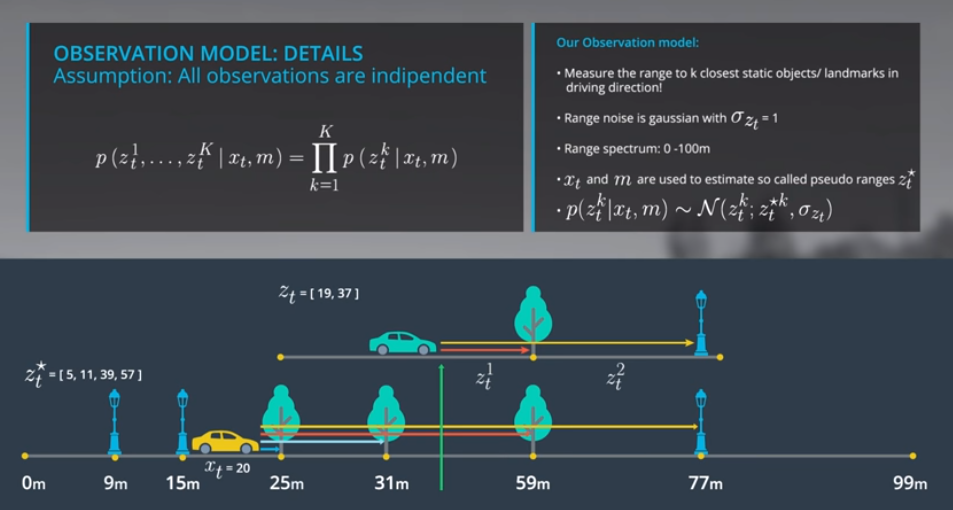
The observation model also varies based on the type of map like dense 2D or 3D Grid Maps, or even sparse feature based maps.

In the 1D example, our sensor measures the distance to the closest n landmarks.



Now our Observation Model will measure the range to the closest static objects in the driving direction. And the range noise is a Gaussian distribution with a standard deviation of 1 meter.

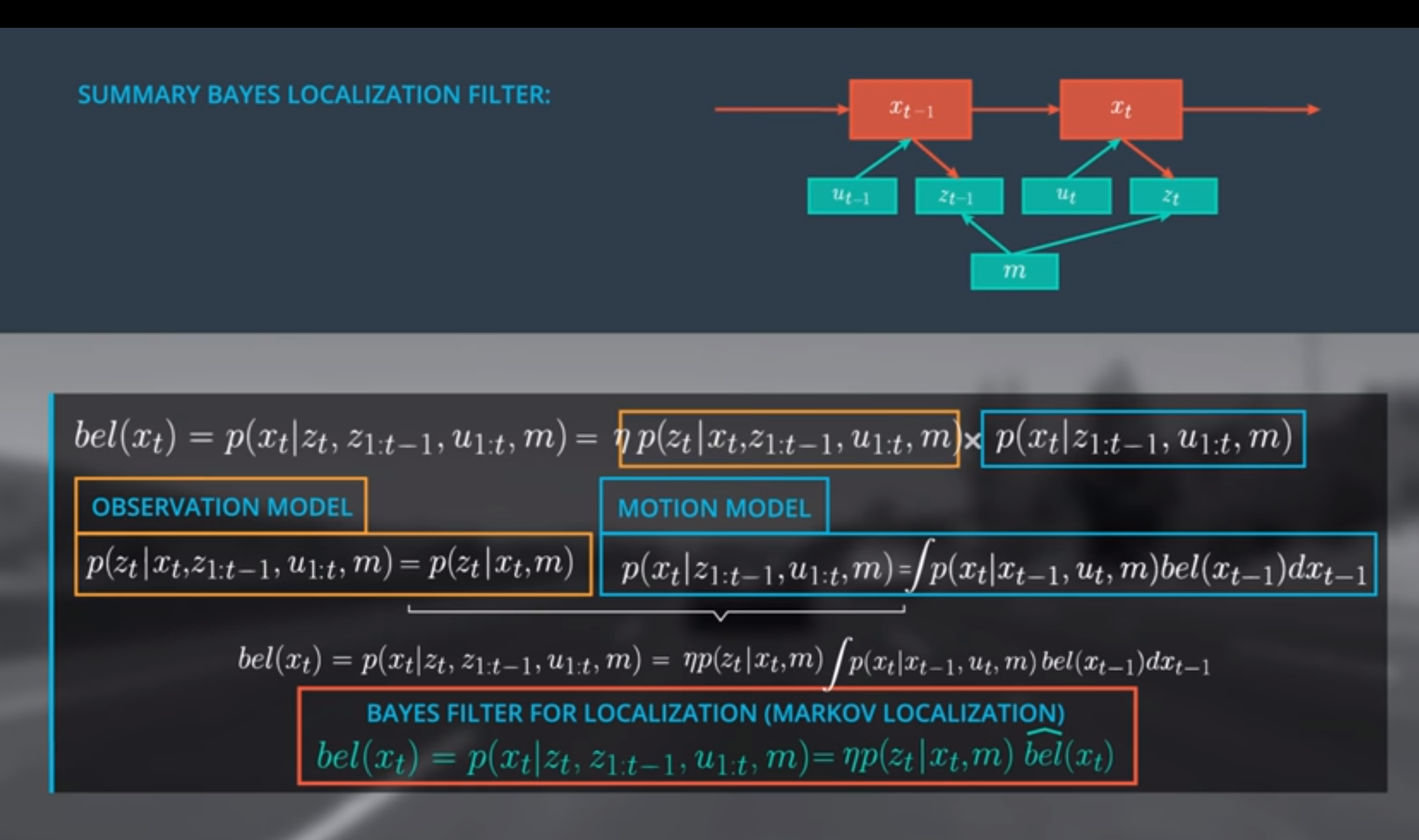
We use xt and m to estimate pseudo ranges, which is another vector the size of the map where each position is the assumption that the car is located (xt=20 for example) would observe the distance to all the other n landmarks.



So each individual observation measurement is.



**Overview:**



**Summary**

