**Markov Localization and the Kidnapped Vehicle Project**

The localization module culminates in the Kidnapped Vehicle Project. In that project our vehicle has been kidnapped and placed in an unknown location. We must leverage our knowledge of localization to determine where our vehicle is. The Kidnapped Vehicle Project relies heavily on the particle filter approach to localization, particularly "Implementation of a Particle Filter," an upcoming lesson. This leaves the question; How does Markov Localization relate to the Kidnapped Vehicle project?

Markov Localization or Bayes Filter for Localization is a generalized filter for localization and all other localization approaches are realizations of this approach, as we'll discuss later on. By learning how to derive and implement (coding exercises) this filter we develop intuition and methods that will help us solve any vehicle localization task, including implementation of a particle filter. We don't know exactly where our vehicle is at any given time, but can approximate it's location. As such, we generally think of our vehicle location as a probability distribution, each time we move, our distribution becomes more diffuse (wider). We pass our variables (map data, observation data, and control data) into the filter to concentrate (narrow) this distribution, at each time step. Each state prior to applying the filter represents our prior and the narrowed distribution represents our Bayes' posterior.

**Bayes' Rule**

If you'd like a reminder about how Bayes' rule works, make sure to go back and watch Sebastian's Bayes' rule video from the Localization Overview lesson!

## Formal Definition of Variables

z\_{1:t}*z*1:*t*​ represents the observation vector from time 0 to t (range measurements, bearing, images, etc.).

u\_{1:t}*u*1:*t*​ represents the control vector from time 0 to t (yaw/pitch/roll rates and velocities).

m*m* represents the map (grid maps, feature maps, landmarks)

x\_t*xt*​ represents the pose (position (x,y) + orientation \theta*θ*)

### Quiz

Given the map, the control elements of the car, and the observations, what is the definition of the posterior distribution for the state x at time t?

Before we dive into deeper into Markov localization, we should review Bayes' Rule. This will serve as a refresher for those familiar with Bayesian methods and we provide some additional resources for those less familiar.

Recall that Bayes' Rule enables us to determine the conditional probability of a state given evidence P(a|b) by relating it to the conditional probability of the evidence given the state P(b|a) in the form of:

which can be rearranged to:

In other words the probability of state a, given evidence b, is the probability of evidence b, given state a, multiplied by the probability of state a, normalized by the total probability of b over all states.

Let's move on to an example to illustrate the utility of Bayes' Rule.

## Bayes' Rule Applied

Let's say we have two bags of marbles, bag 1 and bag 2, filled with two types of marbles, red and blue. Bag 1 contains 10 blue marbles and 30 red marbles, whereas bag 2 contains 20 of each color marble.

If a friend were to choose a bag at random and then a marble at random, from that bag, how can we determine the probability that that marble came from a specific bag? You guessed it - Bayes' Rule!

In this scenario, our friend produces a red marble, in that case, what is the probability that the marble came from bag 1? Rewriting this in terms of Bayes' Rule, our solution becomes:

Let's walk through the process in the following quizzes.

**Bayesian Methods Resources**

* [Sebastian Discusses Bayes Rule](https://classroom.udacity.com/nanodegrees/nd013/parts/30260907-68c1-4f24-b793-89c0c2a0ad32/modules/28233e55-d2e8-4071-8810-e83d96b5b092/lessons/3c8dae65-878d-4bee-8c83-70e39d3b96e0/concepts/487221690923?contentVersion=2.0.0&contentLocale=en-us)
* [More Bayes Rule Content from Udacity](https://classroom.udacity.com/courses/st101/lessons/48703346/concepts/483698470923)
* [Bayes Rule with Ratios](https://betterexplained.com/articles/understanding-bayes-theorem-with-ratios)
* [A Deep Dive into Bayesian Methods, for Programmers](http://greenteapress.com/wp/think-bayes/)

We can apply Bayes' Rule to vehicle localization by passing variables through Bayes' Rule for each time step, as our vehicle moves. This is known as a Bayes' Filter for Localization. We will cover the specific as the lesson continues, but the generalized form Bayes' Filter for Localization is shown below. You may recognize this as being similar to a Kalman filter. In fact, many localization filters, including the Kalman filter are special cases of Bayes' Filter.

Remember the general form for Bayes' Rule:

With respect to localization, these terms are:

1. *P*(*location*∣*observation*): This is P(a|b), the **normalized** probability of a position given an observation (posterior).
2. *P*(*observation*∣*location*): This is P(b|a), the probability of an observation given a position (likelihood)
3. *P*(*location*): This is P(a), the prior probability of a position
4. *P*(*observation*): This is P(b), the total probability of an observation

Without going into detail yet, be aware that  *P*(*location*) is determined by the motion model. The probability returned by the motion model is the product of the transition model probability (the probability of moving from x\_{t-1}*xt*−1​ --> x\_t*xt*​ and the probability of the state x\_{t-1}*xt*−1​.

Over the course of this lesson, you’ll build your own Bayes’ filter. In the next few quizzes, you’ll write code to:

1. Compute Bayes’ rule
2. Calculate Bayes' posterior for localization
3. Initialize a prior belief state
4. Create a function to initialize a prior belief state given landmarks and assumptions

To continue developing our intuition for this filter and prepare for later coding exercises, let's walk through calculations for determining posterior probabilities at several pseudo positions x, for a single time step. We will start with a time step after the filter has already been initialized and run a few times. We will cover initialization of the filter in an upcoming concept.

| **pseudo\_position (x)** | **P(location)** | **P(observation∣location)** | **Raw P(location∣observation)** | **Normalized P(location∣observation)** |
| --- | --- | --- | --- | --- |
| 1 | 1.67E-02 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| 2 | 3.86E-02 | 6.99E-03 | ? | 2.59E-02 |
| 3 | 4.90E-02 | 8.52E-02 | 4.18E-03 | 4.01E-01 |
| 4 | 3.86E-02 | ? | 5.42E-03 | 5.21E-01 |
| 5 | 1.69E-02 | 3.13E-02 | 5.31E-04 | 5.10E-02 |
| 6 | 6.51E-03 | 9.46E-04 | 6.16E-06 | ? |
| 7 | ? | 3.87E-06 | 6.55E-08 | 6.29E-06 |
| 8 | 3.86E-02 | 0.00E+00 | 0.00E+00 | 0.00E+00 |

**Normalized P(location\_observation) vs. Raw P(location|observation):** The **Raw P(location|observation)** is the result prior to dividing by the total probability of P(observation), the P(b) term (denominator) of the generalized Bayes`rule. The **normalized P(location|observation)** is the result of after dividing by P(observation).

Remember the general form for Bayes' Rule:

With respect to localization, these terms are:

1. P(location|observation)*P*(*location*∣*observation*): This is P(a|b), the **normalized** probability of a position given an observation (posterior)
2. P(observation|location)*P*(*observation*∣*location*): This is P(b|a), the probability of an observation given a position (likelihood)
3. P(location)*P*(*location*): This is P(a), the prior probability of a position
4. P(observation)*P*(*observation*): This is P(b), the total probability of an observation

To help develop an intuition for this filter and prepare for later coding exercises, let's walk through the process of initializing our prior belief state. That is, what values should our initial belief state take for each possible position? Let's say we have a 1D map extending from 0 to 25 meters. We have landmarks at x = 5.0, 10.0, and 20.0 meters, with position standard deviation of 1.0 meter. If we know that our car's initial position is at one of these three landmarks, how should we define our initial belief state?

Since we know that we are parked next to a landmark, we can set our probability of being next to a landmark as 1.0. Accounting for a position precision of +/- 1.0 meters, this places our car at an initial position in the range **[4, 6]** (5 +/- 1), **[9, 11]** (10 +/- 1), or **[19, 21]** (20 +/- 1). All other positions, not within 1.0 meter of a landmark, are initialized to 0. We normalize these values to a total probability of 1.0 by dividing by the total number of positions that are potentially occupied. In this case, that is 9 positions, 3 for each landmark (the landmark position and one position on either side). This gives us a value of 1.11E-01 for positions +/- 1 from our landmarks (1.0/9). So, our initial belief state is:

{0, 0, 0, 1.11E-01, 1.11E-01, 1.11E-01, 0, 0, 1.11E-01, 1.11E-01, 1.11E-01, 0, 0, 0, 0, 0, 0, 0, 1.11E-01, 1.11E-01, 1.11E-01, 0, 0, 0, 0}

To reinforce this concept, let's practice with a quiz.

* **map size:** 100 meters
* **landmark positions:** {8, 15, 30, 70, 80}
* **position standard deviation:** 2 meters

Assuming we are parked next to a landmark, answer the following questions about our initial belief state.

**Initial Probability for Position 11**

What is our initial probability (initial belief state) for position 11? If the answer is non-zero, enter it in scientific notation with an accuracy of two decimal places, for example 3.14E-15.

1.11E-01

RESET

**Initial Probability for Position 71**

What is our initial probability (initial belief state) for position 71? If the answer is non-zero, enter it in scientific notation with an accuracy of two decimal places, for example 3.14E-15.

SUBMIT

To determine the initial probability we will divide 1.0 by the total number of positions within 2 meters of a landmark. In this case we have 5 landmarks and a position standard deviation of 2.0 meters. This gives us 5 potentially occupied positions per landmark (the landmark position and 2 each side), yielding 25 potentially occupied positions (5 landmarks \* 5 positions/landmark).

**1.0/25 = 4.00E-02** - remember to enter two decimal places!

Show Solution

In the next concept, we will implement belief state initialization in C++.

Graphical user interface, website

Description automatically generated

We aim to estimate state beliefs bel(x\_t)*bel*(*xt*​) without the need to carry our entire observation history. We will accomplish this by manipulating our posterior p(x\_t|z\_{1:t-1},\mu\_{1:t},m)*p*(*xt*​∣*z*1:*t*−1​,*μ*1:*t*​,*m*), obtaining a recursive state estimator. For this to work, we must demonstrate that our current belief bel(x\_t)*bel*(*xt*​) can be expressed by the belief one step earlier bel(x\_{t-1})*bel*(*xt*−1​), then use new data to update only the current belief. This recursive filter is known as the Bayes Localization filter or Markov Localization, and enables us to avoid carrying historical observation and motion data. We will achieve this recursive state estimator using Bayes Rule, the Law of Total Probability, and the Markov Assumption.

Graphical user interface, website

Description automatically generated

We take the first step towards our recursive structure by splitting our observation vector z\_{1:t}*z*1:*t*​ into current observations z\_t*zt*​ and previous information z\_{1:t-1}*z*1:*t*−1​. The posterior can then be rewritten as p(x\_t|z\_t,z\_{1:t-1},u\_{1:t}, m)*p*(*xt*​∣*zt*​,*z*1:*t*−1​,*u*1:*t*​,*m*).

Graphical user interface, website

Description automatically generated

Now, we apply Bayes' rule, with an additional challenge, the presence of multiple distributions on the right side (likelihood, prior, normalizing constant). How can we best handle multiple conditions within Bayes Rule? As a hint, we can use substitution, where x\_t*xt*​ is a, and the observation vector at time t, is b. Don’t forget to include u*u* and m*m* as well.

### Bayes Rule

P(a \mid b) = \frac{P(b \mid a) \, P(a)}{P(b)}*P*(*a*∣*b*)=*P*(*b*)*P*(*b*∣*a*)*P*(*a*)​

### Quiz

Please apply Bayes Rule to determine the right side of Bayes rule, where the posterior, P(a|b)*P*(*a*∣*b*), is p(x\_t|z\_t,z\_{1:t-1},u\_{1:t},m)*p*(*xt*​∣*zt*​,*z*1:*t*−1​,*u*1:*t*​,*m*)

(A)\frac{p(x\_t|z\_t,z\_{1:t-1},u\_{1:t},m)p(z\_t|x\_t,z\_{1:t-1},u\_{1:t},m)}{p(z\_t|z\_{1:t-1},u\_{1:t},m)}*p*(*zt*​∣*z*1:*t*−1​,*u*1:*t*​,*m*)*p*(*xt*​∣*zt*​,*z*1:*t*−1​,*u*1:*t*​,*m*)*p*(*zt*​∣*xt*​,*z*1:*t*−1​,*u*1:*t*​,*m*)​

(B)\frac{p(z\_t|x\_t,z\_{1:t-1},u\_{1:t},m)p(x\_t|z\_{1:t-1},u\_{1:t},m)}{p(z\_t|z\_{1:t-1},u\_{1:t},m)}*p*(*zt*​∣*z*1:*t*−1​,*u*1:*t*​,*m*)*p*(*zt*​∣*xt*​,*z*1:*t*−1​,*u*1:*t*​,*m*)*p*(*xt*​∣*z*1:*t*−1​,*u*1:*t*​,*m*)​

(C)\frac{p(z\_t|x\_t,z\_{1:t-1},u\_{1:t},m)p(x\_t|z\_{1:t-1},u\_{1:t},m)}{p(x\_t|z\_{1:t-1},u\_{1:t},m)}*p*(*xt*​∣*z*1:*t*−1​,*u*1:*t*​,*m*)*p*(*zt*​∣*xt*​,*z*1:*t*−1​,*u*1:*t*​,*m*)*p*(*xt*​∣*z*1:*t*−1​,*u*1:*t*​,*m*)​

### Markov Assumption for Motion Model Quiz

What do you think about these two assumptions:

(a) Since we (hypothetically) know in which state the system is at time step t-1, the past observations z\_{1:t-1}*z*1:*t*−1​ and controls u\_{1:t-1}*u*1:*t*−1​ would not provide us additional information to estimate the posterior for x\_t*xt*​, because they were already used to estimate x\_{t-1}*xt*−1​. This means, we can simplify p(x\_t|x\_{t-1}, z\_{1:t-1}, u\_{1:t},m)*p*(*xt*​∣*xt*−1​,*z*1:*t*−1​,*u*1:*t*​,*m*) to p(x\_t|x\_{t-1}, u\_t, m)*p*(*xt*​∣*xt*−1​,*ut*​,*m*).

(b) Since u\_t*ut*​ is “in the future” with reference to x\_{t-1}, u\_t*xt*−1​,*ut*​ does not tell us much about x\_{t-1}*xt*−1​. This means the term p(x\_{t-1}|z\_{1:t-1}, u\_{1:t}, m)*p*(*xt*−1​∣*z*1:*t*−1​,*u*1:*t*​,*m*) can be simplified to p(x\_{t-1}|z\_{1:t-1}, u\_{1:t-1}, m)*p*(*xt*−1​∣*z*1:*t*−1​,*u*1:*t*−1​,*m*) .

### Markov Assumption

A Markov process is one in which the conditional probability distribution of future states (ie the next state) is dependent only upon the current state and not on other preceding states. This can be expressed mathematically as:

P(x\_t|x\_{1-t},....,x\_{t-i},...., x\_0) = P(x\_t|x\_{t-1})*P*(*xt*​∣*x*1−*t*​,....,*xt*−*i*​,....,*x*0​)=*P*(*xt*​∣*xt*−1​)

It is important to note that the current state may contain all information from preceding states. That is the case discussed in this lesson.

Graphical user interface

Description automatically generated

We have achieved a very important step towards the final form of our recursive state estimator. Let’s see why. If we rewrite the second term in our integral to split z\_{1-t}*z*1−*t*​ to z\_{t-1}*zt*−1​ and z\_{t-2}*zt*−2​ we arrive at a function that is exactly the belief from the previous time step, namely bel(x\_{t-1})*bel*(*xt*−1​).

Graphical user interface

Description automatically generated

Now we can rewrite out integral as the belief of x\_{t-1}*xt*−1​.

A picture containing text

Description automatically generated

The amazing thing is that we have a recursive update formula and can now use the estimated state from the previous time step to predict the current state at t. This is a critical step in a recursive Bayesian filter because it renders us independent from the entire observation and control history. So in the graph structure, we will replace the previous state terms (highlighted) with our belief of the state at x*x* at t-1*t*−1 (next image).

Text

Description automatically generated

Finally, we replace the integral by a sum over all x\_i*xi*​ because we have a discrete localization scenario in this case, to get the same formula in Sebastian's lesson for localization. The process of predicting x\_t*xt*​ with a previous beliefs (x\_{t-1}*xt*−1​) and the transition model is technically a convolution. If you take a look to the formula again, it is essential that the belief at x\_t = 0*xt*​=0 is initialized with a meaningful assumption. It depends on the localization scenario how you set the belief or in other words, how you initialize your filter. For example, you can use GPS to get a coarse estimate of your location.

Graphical user interface, website

Description automatically generated

# Lesson Breakpoint

Graphical user interface, website

Description automatically generated

Awesome work! Summing up, here is what we have learned so far:

* How to apply the law of total probability by including the new variable x\_{t-1}*xt*−1​.
* The Markov assumption, which is very important for probabilistic reasoning, and allows us to make recursive state estimation without carrying our entire history of information
* How to derive the recursive filter structure.

This is a lesson breakpoint, as it's a good place to pause if you're trying to decide how to tackle this longer lesson. While you'll still use the earlier concepts later on, we'll next be implementing a motion model in C++ and learning how to initialize our localizer.

Whether it's a ten minute break to absorb all the information so far, or coming back tomorrow for more, we'll look forward to seeing you back in the classroom!

Here is a screenshot of the quiz for reference:

Graphical user interface, application

Description automatically generated

quiz screenshot

* **Package name**: Enter com.google.firebase.udacity.greenthumb.
* **App nickname (optional)**: You can leave it blank or add an optional name.
* **Debug signing certificate**: You don’t have to add a signing certificate SHA-1 here because we will not be using Dynamic Links, Invites, or Google Sign-In support. However, if you want to learn more about adding this for future apps you can check out this [link](https://developers.google.com/android/guides/client-auth).

To implement these models in code, we need a function to which we can pass model parameters/values and return a probability. Fortunately, we can use a normalized probability density function (PDF). Let's revisit Sebastian's discussion of this topic.

We have implemented this Gaussian Distribution as a C++ function, normpdf, and will practice using it at the end of this concept. normpdf accepts a value, a parameter, and a standard deviation, returning a probability.

## Additional Resources for Gaussian Distributions

* [Udacity's Statistics Course content on PDF](https://classroom.udacity.com/courses/st095/lessons/86217921/concepts/1020887710923)
* <http://mathworld.wolfram.com/NormalDistribution.html>
* <http://stattrek.com/statistics/dictionary.aspx?definition=Probability_density_function>

Let's practice using normpdf to determine transition model probabilities. Specifically, we need to determine the probability of moving from x\_{t-1}*xt*−1​ --> x\_t*xt*​. The value entered into normpdf will be the distance between these two positions. We will refer to potential values of these positions as pseudo position and pre-pseudo position. For example, if our pseudo position x is 8 and our pre-pseudo position is 5, our sample value will be 3, and our transition will be from x - 3 --> x.

To calculate our transition model probability, pass any difference in distances into normpdf using our control parameter and position standard deviation.

Now we will practice implementing the motion model to determine P(location) for our Bayesian filter. We discussed the derivation of the model in **Recursive Structure**and **Implementation Details for Motion Model**.

Recall that we derived the following recursive structure for the motion model:

\int p(x\_t|x\_{t-1}, u\_t, m)bel(x\_{t-1})dx\_{t-1}∫*p*(*xt*​∣*xt*−1​,*ut*​,*m*)*bel*(*xt*−1​)*dxt*−1​

and that we will implement this in the discretized form:

\sum\limits\_{i} p(x\_t|x\_{t-1}^{(i)}, u\_t, m)bel(x\_{t-1}^{(i)})*i*∑​*p*(*xt*​∣*xt*−1(*i*)​,*ut*​,*m*)*bel*(*xt*−1(*i*)​)

Let's consider again what the summation above is doing - calculating the probability that the vehicle is now at a given location, x\_t*xt*​.

How is the summation doing that? It's looking at each prior location where the vehicle could have been, x\_{t-1}*xt*−1​. Then the summation iterates over every possible prior location, x\_{t-1}^{(1)}...x\_{t-1}^{(n)}*xt*−1(1)​...*xt*−1(*n*)​. For each possible prior location in that list, x\_{t-1}^{(i)}*xt*−1(*i*)​, the summation yields the **total probability** that the vehicle really did start at that prior location **and** that it wound up at x\_t*xt*​.

That now raises the question, how do we calculate the individual probability that the vehicle really did start at that prior location **and** that it wound up at x\_t*xt*​, for each possible starting position x\_{t-1}*xt*−1​?

That's where each individual element of the summation contributes. The likelihood of starting at x\_{t-1}*xt*−1​ and arriving at x\_{t}*xt*​ is simply p(x\_t|x\_{t-1}) \* p(x\_{t-1})*p*(*xt*​∣*xt*−1​)∗*p*(*xt*−1​).

We can say the same thing, using different notation and incorporating all of our knowledge about the world, by writing: p(x\_t|x\_{t-1}^{(i)}, u\_t, m) \* bel(x\_{t-1}^{(i)})*p*(*xt*​∣*xt*−1(*i*)​,*ut*​,*m*)∗*bel*(*xt*−1(*i*)​)

From the equation above we can see that our final position probability is the sum of n discretized motion model calculations, where each calculation is the product of the 'i'th transition probability, p(x\_t|x\_{t-1}^{(i)}, u\_t, m)*p*(*xt*​∣*xt*−1(*i*)​,*ut*​,*m*), and 'i'th belief state, bel(x\_{t-1}^{(i)})*bel*(*xt*−1(*i*)​). Let's try out a single, discreet calculation.

'**i**'**th Motion Model Probability:**

p(x\_t|x\_{t-1}^{(i)}, u\_t, m) \* bel(x\_{t-1}^{(i)})*p*(*xt*​∣*xt*−1(*i*)​,*ut*​,*m*)∗*bel*(*xt*−1(*i*)​)

| **pseudo\_position (x)** | **pre-pseudo\_position** | **delta position** | **P(transition)** | bel(x\_{t-1})*bel*(*xt*−1​) | **P(position)** |
| --- | --- | --- | --- | --- | --- |
| 7 | 1 | 6 | 1.49E-06 | 5.56E-02 | 8.27E-08 |
| 7 | 2 | 5 | 1.34E-04 | 5.56E-02 | 7.44E-06 |
| 7 | 3 | 4 | 4.43E-03 | 5.56E-02 | 2.46E-04 |
| 7 | 4 | ? | 5.40E-02 | 0.00E+00 | 0.00E+00 |
| 7 | 5 | 2 | ? | 0.00E+00 | 0.00E+00 |
| 7 | 6 | 1 | 3.99E-01 | 0.00E+00 | 0.00E+00 |
| 7 | 7 | 0 | 2.42E-01 | ? | 1.66E-03 |
| 7 | 8 | -1 | 5.40E-02 | 1.79E-03 | ? |

In the next concept we will implement the motion model in C++.

**Reference Equations**

* **Discretized Motion Model:**

\sum\limits\_{i} p(x\_t|x\_{t-1}^{(i)}, u\_t, m)bel(x\_{t-1}^{(i)})*i*∑​*p*(*xt*​∣*xt*−1(*i*)​,*ut*​,*m*)*bel*(*xt*−1(*i*)​)

* **Transition Model:**

p(x\_t|x\_{t-1}^{(i)}, u\_t, m)*p*(*xt*​∣*xt*−1(*i*)​,*ut*​,*m*)

* '**i**'**th Motion Model Probability:**

p(x\_t|x\_{t-1}^{(i)} u\_t, m) \*bel(x\_{t-1}^{(i)})*p*(*xt*​∣*xt*−1(*i*)​*ut*​,*m*)∗*bel*(*xt*−1(*i*)​)

Now that we have manually calculated each step for determining the motion model probability, we will implement these steps in a function. The starter code below steps through each position x, calls the motion\_model function and prints the results to stdout. To complete this exercise fill in the motion\_model function which will involve:

* For each x\_{t}*xt*​. :
  + Calculate the transition probability for each potential value x\_{t-1}*xt*−1​
  + Calculate the discrete motion model probability by multiplying the transition model probability by the belief state (prior) for x\_{t-1}*xt*−1​
* Return total probability (sum) of each discrete probability

### Observation Model

p(z\_t|x\_t,z\_{1:t-1},u\_{1:t},m)*p*(*zt*​∣*xt*​,*z*1:*t*−1​,*u*1:*t*​,*m*)

### Motion Model

p(x\_t|z\_{1:t-1},u\_{1:t},m)*p*(*xt*​∣*z*1:*t*−1​,*u*1:*t*​,*m*)

#### Quiz: Simplifying the Observation Model



The Markov assumption can help us simplify the observation model. Recall that the Markov Assumption is that the next state is dependent only upon the preceding states and that preceding state information has already been used in our state estimation. As such, we can ignore terms in our observation model prior to x\_t*xt*​ since these values have already been accounted for in our current state and assume that t is independent of previous observations and controls.

Graphical user interface, website

Description automatically generated

With these assumptions we simplify our posterior distribution such that the observations at t are dependent only on x at time t and the map.

Graphical user interface, website

Description automatically generated

Since z\_t*zt*​ can be a vector of multiple observations we rewrite our observation model to account for the observation models for each single range measurement. We assume that the noise behavior of the individual range values z\_t^1*zt*1​ to z\_t^k*ztk*​ is independent and that our observations are independent, allowing us to represent the observation model as a product over the individual probability distributions of each single range measurement. Now we must determine how to define the observation model for a single range measurement.

Timeline

Description automatically generated

In general there exists a variety of observation models due to different sensor, sensor specific noise behavior and performance, and map types. For our 1D example we assume that our sensor measures to the n closest objects in the driving direction, which represent the landmarks on our map. We also assume that observation noise can be modeled as a Gaussian with a standard deviation of 1 meter and that our sensor can measure in a range of 0 – 100 meters.

To implement the observation model we use the given state x\_t*xt*​, and the given map to estimate pseudo ranges, which represent the true range values under the assumption that your car would stand at a specific position x\_t*xt*​, on the map. For example, if our car is standing at position 20 it would make use x\_t*xt*​, and m to make pseudo range (z\_t^\**zt*∗​) observations in the order of the first landmark to the last landmark or 5, 11, 39, and 57 meters. Compared to our real observations (z\_t*zt*​ = [19, 37]) the position x\_t*xt*​, = 20 seems unlikely and our observation would rather fit to a position around 40.

Based on this example the observation model for a single range measurement is defined by the probability of the following normal distribution p(z\_t^k|x\_t )\tilde\ N(z\_t^k,z\_t^{\*k},\sigma z\_t)*p*(*ztk*​∣*xt*​) ~*N*(*ztk*​,*zt*∗*k*​,*σzt*​) where z\_t^{\*k}*zt*∗*k*​ is the mean. This insight will ultimately allow us to implement the observation model in c++.

Graphical user interface, website

Description automatically generated

We have accomplished a lot in this lesson.

* [Starting with the generalized form of Bayes Rule](https://classroom.udacity.com/nanodegrees/nd013/parts/40f38239-66b6-46ec-ae68-03afd8a601c8/modules/2c318113-724b-4f9f-860c-cb334e6e4ad7/lessons/47f9b7a1-317f-4fab-88d3-bb3ce215d575/concepts/d56d2238-9528-46c8-b1e1-ad086a73f3bb) we expressed our posterior, the belief of x at t as \eta*η* (normalizer) multiplied with the observation model and the motion model.
* [We simplified the observation model](https://classroom.udacity.com/nanodegrees/nd013/parts/40f38239-66b6-46ec-ae68-03afd8a601c8/modules/2c318113-724b-4f9f-860c-cb334e6e4ad7/lessons/47f9b7a1-317f-4fab-88d3-bb3ce215d575/concepts/17fdc941-ea11-4438-b8b7-4dedf70283fd) using the Markov assumption to determine the probability of z at time t, given only x at time t, and the map.
* We expressed the [motion model](https://classroom.udacity.com/nanodegrees/nd013/parts/40f38239-66b6-46ec-ae68-03afd8a601c8/modules/2c318113-724b-4f9f-860c-cb334e6e4ad7/lessons/47f9b7a1-317f-4fab-88d3-bb3ce215d575/concepts/530a769e-637e-40ce-ab73-ad5293a24d88) as a recursive state estimator using the Markov assumption and the law of total probability, resulting in a model that includes our belief at t – 1 and our transition model.
* Finally we derived the general Bayes Filter for Localization (Markov Localization) by expressing our belief of x at t as a simplified version of our original posterior expression (top equation), \eta*η* multiplied by the simplified observation model and the motion model. Here the motion model is written as \hat{bel}*bel*^, a prediction model.

Graphical user interface

Description automatically generated

The Bayes Localization Filter dependencies can be represented as a graph, by combining our sub-graphs. To estimate the new state x at t we only need to consider the previous belief state, the current observations and controls, and the map.

Graphical user interface

Description automatically generated

It is a common practice to represent this filter without the belief x\_t*xt*​ and to remove the map from the motion model. Ultimately we define bel(x\_t)*bel*(*xt*​) as the following expression.

### Bayes Filter for Localization (Markov Localization)

bel(x\_t) = p(x\_t|z\_t,z\_{1:t-1},\mu\_{1:t},m) = \eta \*p(z\_t|x\_t,m) \hat{bel}(x\_t)*bel*(*xt*​)=*p*(*xt*​∣*zt*​,*z*1:*t*−1​,*μ*1:*t*​,*m*)=*η*∗*p*(*zt*​∣*xt*​,*m*)*bel*^(*xt*​)

Graphical user interface, application

Description automatically generated

The image above sums up the core achievements of this lesson.

* The Bayes Localization Filter Markov Localization is a general framework for recursive state estimation.
* That means this framework allows us to use the previous state (state at t-1) to estimate a new state (state at t) using only current observations and controls (observations and control at t), rather than the entire data history (data from 0:t).

Graphical user interface

Description automatically generated

* The motion model describes the prediction step of the filter while the observation model is the update step.
* The state estimation using the Bayes filter is dependent upon the interaction between prediction (motion model) and update (observation model steps) and all the localization methods discussed so far are realizations of the Bayes filter.
* In the next few sections we will learn how to estimate pseudo ranges, calculate the observation model probability, and complete the implementation of the observation model in C++.

We will complete our Bayes' filter by implementing the observation model. The observation model uses pseudo range estimates and observation measurements as inputs. Let's recap what is meant by a pseudo range estimate and an observation measurement.

For the figure below, the top 1d map (green car) shows our observation measurements. These are the distances from our actual car position at time t, to landmarks, as detected by sensors. In this example, those distances are 19m and 37m.

The bottom 1d map (yellow car) shows our pseudo range estimates. These are the distances we would expect given the landmarks and assuming a given position x at time t, of 20m. In this example, those distances are 5, 11, 39, and 57m.

Diagram

Description automatically generated

The observation model will be implemented by performing the following at each time step:

* Measure the range to landmarks up to 100m from the vehicle, in the driving direction (forward)
* Estimate a pseudo range from each landmark by subtracting pseudo position from the landmark position
* Match each pseudo range estimate to its closest observation measurement
* For each pseudo range and observation measurement pair, calculate a probability by passing relevant values to norm\_pdf: norm\_pdf(observation\_measurement, pseudo\_range\_estimate, observation\_stdev)
* Return the product of all probabilities

Why do we multiply all the probabilities in the last step? Our final signal (probability) must reflect all pseudo range, observation pairs. This blends our signal. For example, if we have a high probability match (small difference between the pseudo range estimate and the observation measurement) and low probability match (large difference between the pseudo range estimate and the observation measurement), our resultant probability will be somewhere in between, reflecting the overall belief we have in that state.

Let's practice this process using the following information and norm\_pdf.

* **pseudo position:** x = 10m
* **vector of landmark positions from our map:** [6m, 15m, 21m, 40m]
* **observation measurements:** [5.5m, 11m]
* **observation standard deviation:** 1.0m