Tracking

Overview

Tracking is the computational process of keeping track of objects in an image over time.

1 Tracking with dynamics

Given a model of expected motion, *predict* where objects will occur in the next frame, even before seeing the image.

- This allows us to restrict our search space for the object
- Get improved estimates since measurement noise is reduced by trajectory smoothness.

Assumption in this model

- Continuous motion patterns
    - Objects do not disappear and appear in different places.
    - Camera doesn't move instantaneously.

2 Tracking vs detection

- The idea of using prediction is the difference between tracking and detection.
- **Detection**:
    - We detect the object independently in each frame.
- **Tracking**
    - We ***predict*** the new location of the object in the next frame using estimated dynamics.
    - Then we ***update*** based on measurements.

3 Tracking as inference

We assume a **hidden state (X)**: The true parameters we care about.
We have a **measurement (Y)**: A noisy observation of the underlying hidden state using a sensor like a camera.
At each timestep t, the state changes (from X_{t-1} to X_t) and we get a new observation Y_t.

Goal: Recover or estimate the distribution or the most likely state X_t given

- All observations so far
- Knowledge about dynamics of state transition (motion model)

Steps for tracking:

- **Prediction**: Compute the next state given past measurements
  $P(X_t|Y_0, Y_1, \dots Y_{t-1})$ (prior)

- **Correction**: Compute an updated estimate from prediction and measurement.
  $P(X_t|Y_0, Y_1, \dots Y_{t-1}, Y_t)$ (posterior)

Tracking is thus the process of propagating the ***posterior*** distribution of state given measurements across time.

Why probability for measurement?

- X and Y could be different quantities
- Noisy observations give us unreliable Y
- Y maybe missing or invalid for some timesteps.

4 Simplifying Assumptions

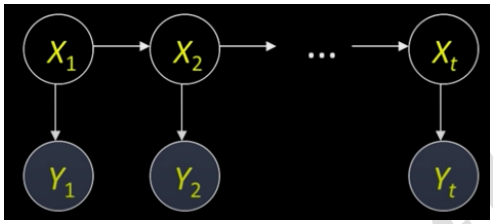
- **Markov Assumption**: For the hidden state, only the immediate past matters

  $P(X_t|X_0, X_1, \dots, X_{t-1}) \approx P(X_t|\, X_{t-1})$ (called the ***Dynamics model***)

- **Hidden Markov Model**: Measurements depend only on the current state.

  $P(Y_t|X_0, Y_0, \dots, X_{t-1}, Y_{t-1}, X_t) \approx P(Y_t|\, X_t)$ (called the ***Observation Model***)

  This can be represented as follows:



5 Tracking as induction

Tracking can also be thought of as an induction process.

- **Base Case**:
  - Assume we have an initial prior that predicts state in the absence of any evidence: $P(X_0)$
  - At first frame, correct this prior given $Y_0 = y_0$

- Given corrected estimate for time t,
  - Predict for time t+1
  - Correct for time t+1

6 Prediction Step

Given: $P(X_{t-1}|Y_0, \dots, Y_{t-1})$, i.e. we have distribution of prior state and all prior measurements.

Find: $P(X_t|Y_0, \dots, Y_{t-1})$ i.e. find distribution of current state

$$P(X_t|Y_0, \dots, Y_{t-1}) = \int P(X_t, X_{t-1}|\, Y_0, \dots, Y_{t-1})\, dX_{t-1}$$

Using ($P(A) = \int P(A,B)\, dB$, marginalization)

$$= \int P(X_t \mid X_{t-1}, Y_0, \dots, Y_{t-1}).\, P(X_{t-1}|\, Y_0, \dots, Y_{t-1})\, dX_{t-1} \ ,$$

Using $P(A,B) = P(A|B).P(B)$

$$= \int P(X_t|\, X_{t-1})\, .\, P(X_{t-1}|\, Y_0, \dots, Y_{t-1})\, dX_{t-1}$$

Using Markov assumption, dynamics model

$$P(X_t \mid y_0,\dots,y_{t-1}) = \int \underbrace{P(X_t \mid X_{t-1})}_{\text{dynamics model}} \underbrace{P(X_{t-1} \mid y_0,\dots,y_{t-1})}_{\text{corrected estimate from previous step}} dX_{t-1}$$

7 Correction Step

Given: $P(X_t|\, Y_0, \dots, Y_{t-1})$ i.e the prediction or prior and Y_t i.e. the latest measurement.

Find: $P(X_t|Y_0, \dots, Y_{t-1}, Y_t)$ i.e the corrected value or posterior.

$$P(X_t|Y_0, \dots, Y_{t-1}, Y_t) = \frac{P(Y_t \mid X_t, Y_0,\dots,Y_{t-1})\, .P(X_t|\, Y_0,\dots,Y_{t-1})}{P(Y_t \mid Y_0,\dots,Y_{t-1})}$$

Using Bayes rule, $P(A|B) = \frac{P(B|A).\ P(A)}{P(B)}$

$$= \frac{P(Y_t|X_t).P(X_t|\, Y_0,\dots,Y_{t-1})}{P(Y_t \mid Y_0,\dots,Y_{t-1})}$$

Using HMM assumption, Observation model

$$= \frac{P(Y_t|X_t).P(X_t|\, Y_0,\dots,Y_{t-1})}{\int P(Y_t|X_t).\ P(X_t \mid Y_0,\dots,Y_{t-1})\, dX_t}$$

Conditioning on X_t and using Bayes rule. (Normalization)

$$P(X_t \mid y_0,\dots,y_{t-1}, y_t) = \frac{\overbrace{P(y_t \mid X_t)}^{\text{observation model}} \overbrace{P(X_t \mid y_0,\dots,y_{t-1})}^{\text{predicted estimate}}}{\int P(y_t \mid X_t) P(X_t \mid y_0,\dots,y_{t-1})\, dX_t}$$

8 Kalman Filter

The Kalman filter is a special case of the tracking model described earlier, that makes the following assumptions:

Linear dynamics model:

- The dynamics model undergoes linear transformation and has some Gaussian noise.
- Formally, the state at time t, will be distributed according to a normal distribution whose mean is a linear transformation of the state at t-1 and variance is some gaussian noise.

$$\mathbf{x}_t \sim N\left(D_t \mathbf{x}_{t-1}, \Sigma_{d_t}\right)$$

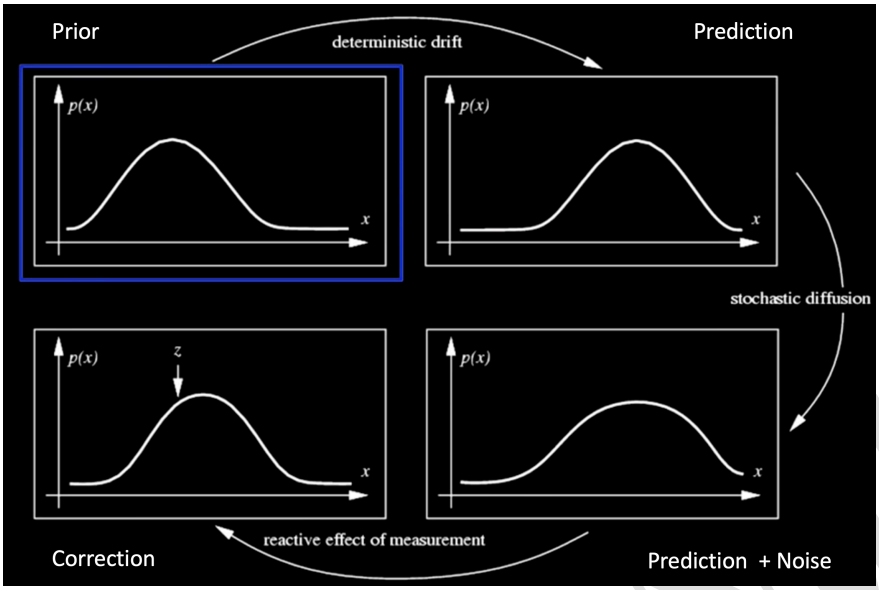
Linear Measurements model:

- In the observation model, measurement is linearly transformed and has some gaussian noise.
- Formally, the measurement at time t, is distributed according to a normal distribution whose mean is a linear transformation of the **state** at time t and variance is some gaussian noise.

$$\mathbf{y}_t \sim N\left(M_t \mathbf{x}_t, \Sigma_{m_t}\right)$$

Definition:

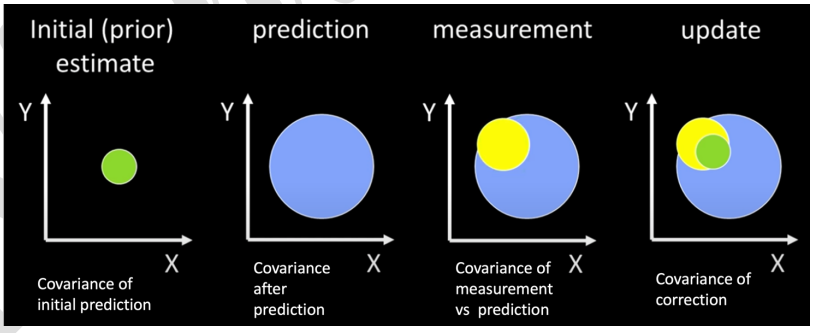
- The Kalman filter is a method of tracking linear dynamical systems with Gaussian noise.
- With a Kalman filter, both the predicted states and the corrected states are Gaussian.
- We only need to maintain the mean and covariance of the states to run the filter.



The above describes the states of a Kalman filter in terms of Gaussians:

- First, we have our initial belief or the prior.
- We then make a prediction for the next time step. This shifts our Gaussian to the right.
- Stochastic diffusion represents the addition of noise to our prediction. Since there is uncertainty, the noise flattens the curve indicating a wider range of uncertainty.
- Once we make our measurement, the correction lands in between the prior and the prediction. Also, since more information always reduces uncertainty, the gaussian is now steeper (less variance).

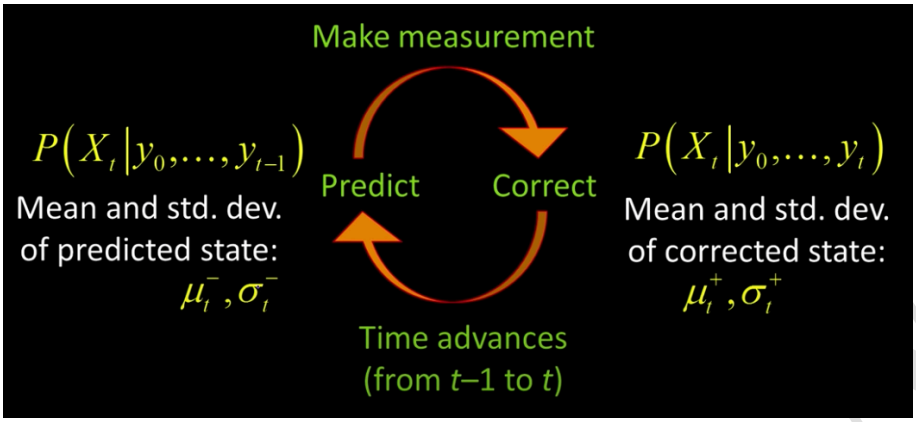
Alternatively, we can compare covariance (or uncertainty) of the steps as follows:



Here, we see that covariance of the correction is smaller than both the covariance of the measurement as well as that of the prediction. This just affirms the fact that more information always reduces uncertainty.

8.1 Notation to formalize the filter

μ_t^-, σ_t^- indicate the prior state or predicted state before taking a measurement.
μ_t^+, σ_t^+ indicate the state after applying the correction



8.2 Kalman Prediction: 1D

Let the predicted state with noise be:

$$X_t \sim N(d.X_{t-1}, \sigma_d^2)$$

From our dynamics model we have:

$$P(X_t|Y_0,\ldots,Y_{t-1}) = \int P(X_t|X_{t-1}).P(X_{t-1}|Y_0,\ldots,Y_{t-1})\,dX_{t-1}$$
$$= N(\mu_t^-, (\sigma_t^-)^2) \text{ , Since it is a Gaussian}$$

The above gives us:

- mean at time t = dynamics multiplier d * corrected mean at (t - 1)

$$\mu_t^- = d.\mu_{t-1}^+$$

- variance at t = d^2 * corrected variance at (t - 1) + additional noise

$$(\sigma_t^-)^2 = (d.\sigma_{t-1}^+)^2 + \sigma_d^2$$

8.3 Kalman Correction: 1D

Let the measurement with noise be:

$$Y_t \sim N(m.X_t, \sigma_m^2)$$

The predicted state from above is:

$$P(X_t|Y_0, \dots, Y_{t-1}) = N\ (\mu_t^-, (\sigma_t^-)^2)$$

From the observation model, the corrected distribution will be:

$$P(X_t|Y_0, \dots, Y_{t-1}, Y_t) = \ N\ (\mu_t^+, (\sigma_t^+)^2)$$

The above gives us:

$$\mu_t^+ = \mu_t^- + k.(\frac{Y_t}{m} - \ \mu_t^-)$$

$$(\sigma_t^+)^2 = (1-k).(\sigma_t^-)^2$$

where $k = \frac{(\sigma_t^-)^2}{\frac{\sigma_m^2}{m^2}+(\sigma_t^-)^2}$ is the Kalman Gain, $0 \leq k \ \leq 1$

Note:

- $\frac{Y_t}{m}$ is the measurement guess of x
- μ_t^- is the prediction of x
- $\frac{\sigma_m^2}{m^2}$ is the variance of x just computed from the measurement
- $(\sigma_t^-)^2$ is the variance of prediction

8.4 Intuition

- What if there is no prediction uncertainty?

  $\Rightarrow (\sigma_t^-)^2 = 0$

  $\Rightarrow k = 0$

  $\Rightarrow \mu_t^+ = \mu_t^- , (\sigma_t^+)^2 = 0$

  i.e. best update is the prediction. No correction or measurement does not matter.
  So, if we are sure of our prediction, ***ignore the measurement***.

- What if there is no measurement uncertainty?

  $\Rightarrow \sigma_m^2 = 0$

  $\Rightarrow k = 1$

  $\Rightarrow \mu_t^+ = \frac{Y_t}{m} , (\sigma_t^+)^2 = 0$

  i.e. best update is the measurement. Prediction does not matter.
  So, if we are sure of our measurement, ***ignore the prediction***.

8.5 N-Dimensional Kalman Filter

In 1 Dimension, we have:

Prediction

$$\mu_t^- = d.\mu_{t-1}^+$$

$$(\sigma_t^-)^2 = (d.\sigma_{t-1}^+)^2 + \sigma_d^2$$

Correction\Measurement update

$$\mu_t^+ = \mu_t^- + k.(\frac{Y_t}{m} - \mu_t^-)$$

$$(\sigma_t^+)^2 = (1-k).(\sigma_t^-)^2$$

where $k = \frac{(\sigma_t^-)^2}{\frac{\sigma_m^2}{m^2}+(\sigma_t^-)^2}$ is the Kalman Gain

In N-Dimensions, we use the vector notation to get:

Prediction

$$X_t^- = D_t.X_{t-1}^+$$
$$\Sigma_t^- = D_t\Sigma_{t-1}^+ D_t^T + \Sigma_{d_t}$$

Where,

$D_t \rightarrow$ State Transition matrix

$\Sigma_{d_t} \rightarrow$ Covariance matrix, process noise

Correction\Measurement update

$$X_t^+ = X_t^- + K_t(Y_t - M_t X_t^-)$$
$$\Sigma_t^+ = (I - K_t M_t)\Sigma_t^-$$

Where,

$$K_t = \Sigma_t^- M_t^T (M_t\Sigma_t^- M_t^T + \Sigma_{m_t})^{-1}$$

$M_t \rightarrow$ Measurement matrix

$Y_t \rightarrow$ Measurement Vector

$\Sigma_{m_t} \rightarrow$ Covariance matrix, Measurement Noise

8.6 Pros and cons

Pros

- Simple updates, compact, efficient

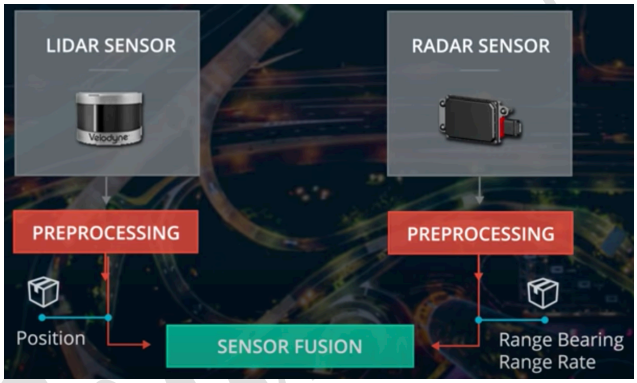
Cons

- Unimodal distribution, only single hypothesis (only Gaussian)
    - Note: Gaussians are unimodal
- Restricted class of motion. Only linear models.
    - "Extended Kalman Filter" solves for non-linear models by approximating to linear.

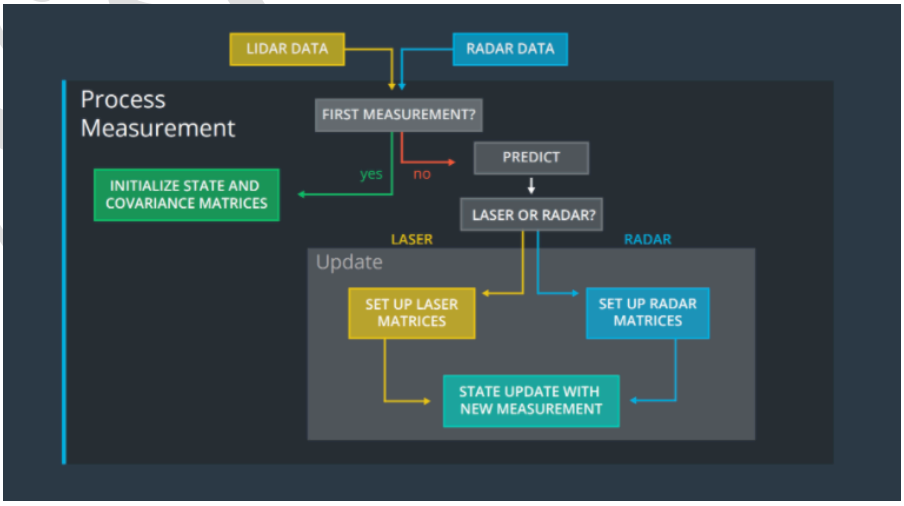
9 Sensor Fusion using Kalman Filters

- So far we have seen the use of Kalman filters using
  - Measurement from a single sensor.
  - Linear model
- Its also possible to use Kalman filters with
  - Multiple measurements from different sensors.
  - Use a non-linear model (extended Kalman filter)
  - Use a mix of linear and non-linear models.

LIDARs give us position with high accuracy, but we cannot sense speed.
RADARs give us radial velocity, but at a lower spatial resolution.
Thus, we can combine these measurements to get better sense of position and velocity.



A high-level overview of sensor fusion is as follows:



- We have 2 sensors, a RADAR and a LIDAR. These are used to estimate the state (position and velocity) of an object (say pedestrian) on the road.
- Whenever new measurements are received, the following are done:
    - **Prediction**
        - In this step, we predict the state and covariance for the next time step.
        - We use the constant velocity model.
    - **Measurement update\correction**
        - This step depends on the sensor type.
            - RADAR
                - Measurements are non-linear
                - Apply an ***Extended Kalman Filter***
            - LIDAR
                - Measurements are linear
                - Apply the ***standard Kalman filter***

9.1 Prediction Step

Let the predicted state be, $X_t \sim N(D_t.X_{t-1}, \Sigma_{d_t})$

Where,

$D_t \rightarrow$ State Transition matrix

$\Sigma_{d_t} \rightarrow$ Process noise

Here, we will use 2D coordinates for position and velocity.

This gives us:

State Vector: $X_t = \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}$,

Where,

Position at time t, $p_t = p_{t-1} + (\Delta t)v_{t-1} + \varepsilon$

Velocity at time t, $v_t = v_{t-1} + \xi$

This can be re-written as:

$$X_t = D_t.X_{t-1} + noise = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix} + \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \xi_x \\ \xi_y \end{bmatrix},$$

The noise, referred to as motion noise or process noise can be defined by taking acceleration into consideration as well. So, we can re-write our state motion model to be:

Position at time t, $p_t = p_{t-1} + (\Delta t)v_{t-1} + \frac{a_{t-1}(\Delta t)^2}{2}$

Velocity at time t, $v_t = v_{t-1} + a_{t-1}\Delta t$

This gives us:

$$X_t = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix} + \begin{bmatrix} \frac{a_x(\Delta t)^2}{2} \\ \frac{a_y(\Delta t)^2}{2} \\ a_x \Delta t \\ a_y \Delta t \end{bmatrix},$$

$$noise, \nu = \begin{bmatrix} \frac{a_x(\Delta t)^2}{2} \\ \frac{a_y(\Delta t)^2}{2} \\ a_x \Delta t \\ a_y \Delta t \end{bmatrix} = \begin{bmatrix} \frac{(\Delta t)^2}{2} & 0 \\ 0 & \frac{(\Delta t)^2}{2} \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} . \begin{bmatrix} a_x \\ a_y \end{bmatrix} = G . \begin{bmatrix} a_x \\ a_y \end{bmatrix} = Ga$$

Δt is computed at each timestep, so G is not a random variable.
The acceleration however can be considered to be a random variable.
we assume it to be:

$$a_x \sim N(0, \Sigma_{a_x})$$
$$a_y \sim N(0, \Sigma_{a_y})$$

Also, we assume the noise is a Gaussian with 0 mean and covariance Σ_{d_t}.

$$noise,\ \nu \sim N(0, \Sigma_{d_t})$$

This gives us:

$$\Sigma_{d_t} = E[\nu\nu^T] \text{ , (since } cov(X,Y) = E[X,Y] - \mu_x.\mu_y)$$
$$= E[Gaa^T G^T]$$
$$= GE[aa^T]G^T$$
$$= G\Sigma_{a_t} G^T,$$

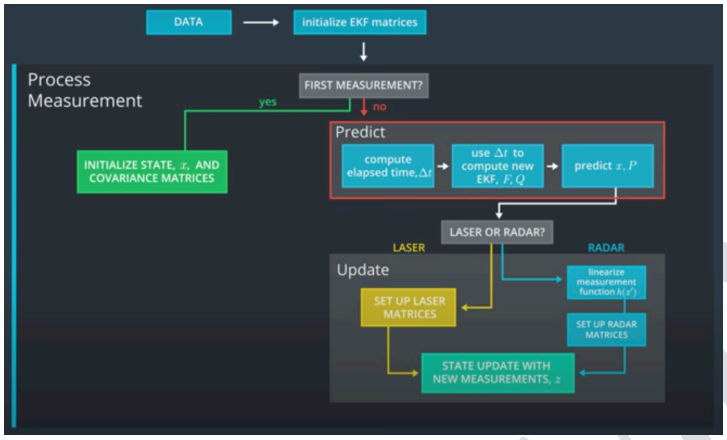
where Σ_{a_t} is the variance of the acceleration random variable

$$\Sigma_{a_t} = \begin{bmatrix} \Sigma_{a_x} & 0 \\ 0 & \Sigma_{a_y} \end{bmatrix}$$

Thus, we get process noise,

$$\Sigma_{d_t} = \begin{bmatrix} \frac{(\Delta t)^2}{2} & 0 \\ 0 & \frac{(\Delta t)^2}{2} \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} \Sigma_{a_x} & 0 \\ 0 & \Sigma_{a_y} \end{bmatrix} \begin{bmatrix} \frac{(\Delta t)^2}{2} & 0 \\ 0 & \frac{(\Delta t)^2}{2} \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}^T$$

9.1.1 Computing the prediction:



1. Compute Δt

2. Compute state transition matrix, D_t

$$D_t = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. Compute Process covariance matrix, Σ_{d_t}

$$\Sigma_{d_t} = \begin{bmatrix} \frac{(\Delta t)^2}{2} & 0 \\ 0 & \frac{(\Delta t)^2}{2} \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} \Sigma_{a_x} & 0 \\ 0 & \Sigma_{a_y} \end{bmatrix} \begin{bmatrix} \frac{(\Delta t)^2}{2} & 0 \\ 0 & \frac{(\Delta t)^2}{2} \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}^T$$

   a. **Note**: Σ_{a_x} and Σ_{a_y} are either provided or assumed.

4. Compute **State Transition**

$$X_t^- = D_t . X_{t-1}^+$$

NOTE:

The first measurement, X_{-1}^+ will need to be converted to cartesian coordinates if measurement came from RADAR:

(see section on RADAR for general details)

$$X_{t-1}^+ = \begin{bmatrix} \rho \,.\cos(\varphi) \\ \rho \,.\sin(\varphi) \\ \dot{\rho} \,.\cos(\varphi) \\ \dot{\rho} \,.\sin(\varphi) \end{bmatrix}$$

5. Compute **Prediction uncertainty**

$$\Sigma_t^- = D_t \Sigma_{t-1}^+ D_t^T + \Sigma_{d_t}$$

9.2 Measurement Update\Correction Step

9.2.1 LIDAR (Laser): Standard Kalman Filter

While taking LADAR measurements, we get a point cloud using which objects are detected and we then get the coordinates of the object.

Let the measurement be, $Z_t \sim N(M_t.X_t, \Sigma_{m_t})$

Where,

$M_t \rightarrow$ Measurement matrix

$\Sigma_{m_t} \rightarrow$ Measurement noise

Here, we will use 2D coordinates for position.

This gives us:

Measurement Vector: $Z_t = \begin{bmatrix} p_x \\ p_y \end{bmatrix}$,

Now, Z has only position, but X has both position and velocity.

So, M needs to transform X such that velocity is ignored.

This gives us:

Measurement vector, $Z_t = M_t X_t + noise = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} . \begin{bmatrix} \grave{p_x} \\ \grave{p_y} \\ \grave{v_x} \\ \grave{v_y} \end{bmatrix} + \text{noise}$

From the above we get:

Measurement Matrix, $M_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$

For the laser measurements, we assume each measurement is affected by some noise.
we assume it to be:

$$p_x \sim N(0, \Sigma_{p_x})$$
$$p_y \sim N(0, \Sigma_{p_y})$$

Also, we assume the measurement noise is a Gaussian with 0 mean and covariance Σ_{m_t}.

$$noise,\ \omega \sim N(0, \Sigma_{m_t})$$

This gives us:

Measurement Noise, $\Sigma_{m_t} = E[\omega\omega^T]$

$$= \begin{bmatrix} \Sigma_{p_x} & 0 \\ 0 & \Sigma_{p_y} \end{bmatrix},$$

(off diag is 0 because we assume p_x and p_y are not correlated)

9.2.2 Compute Measurement Update or correction

Inputs are the following:

- State prediction, X_t^-
- Prediction uncertainty, Σ_t^-
- Measurement vector, Z_t
- Measurement noise, Σ_{m_t}

1. We know the Measurement Matrix,

$$M_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

2. Compute Kalman Gain,

$$K_t = \Sigma_t^- \mathrm{M}_t^T \ (M_t \Sigma_t^- \mathrm{M}_t^T + \Sigma_{m_t})^{-1}$$

3. Compute **updated measurement**

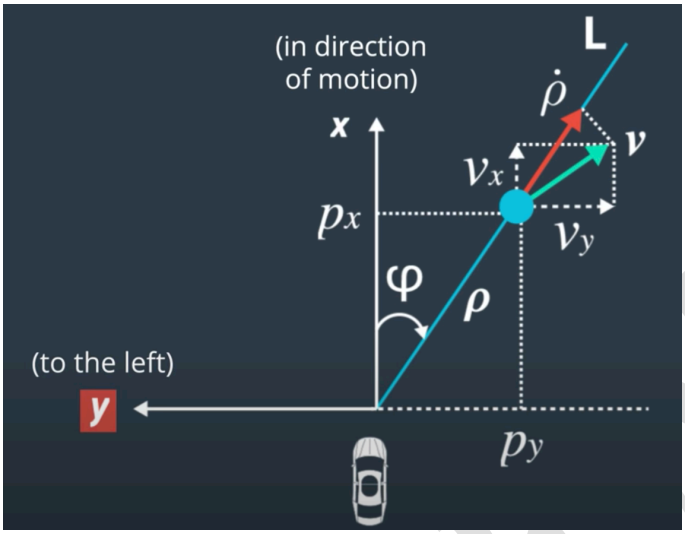
$$X_t^+ = X_t^- + K_t(Z_t - M_t X_t^-)$$

4. Compute **update uncertainty**

$$\Sigma_t^+ = (I - K_t M_t)\Sigma_t^-$$

9.2.3 RADAR: Extended Kalman Filter

RADAR gives us radial velocity.



We get the following:

- Coordinates: X goes up. Y goes to the left.
- **Range**, ρ (rho): Radial distance from origin to a point (p_x, p_y)
- **Bearing**, φ (phi): Angle between ρ and X-axis
- **Radial velocity**, $\dot{\rho}$ $(rho\ dot)$: Rate of change of ρ

Now, we have the following relationships between cartesian and polar coordinates:

$$\rho = \sqrt{p_x^2 + p_y^2}$$

$$\varphi = \arctan\left(\frac{p_x}{p_y}\right)$$

$$\dot{\rho} = \frac{p_x v_x + p_y v_y}{\sqrt{p_x^2 + p_y^2}}$$

Clearly, the above are non-linear relationships.

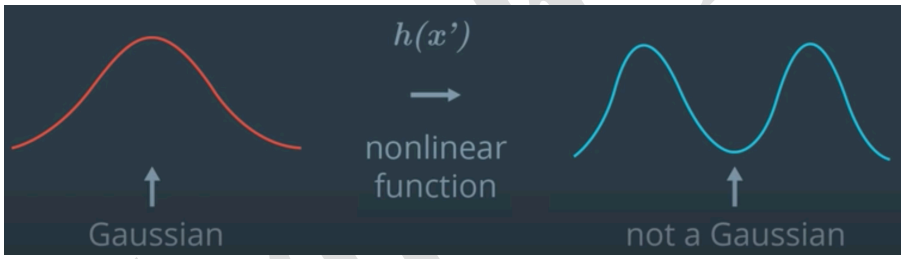
Let the measurement be, $Z_t \sim N(M_t . X_t, \Sigma_{m_t})$

From the prediction step we have, $X_t = \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}$

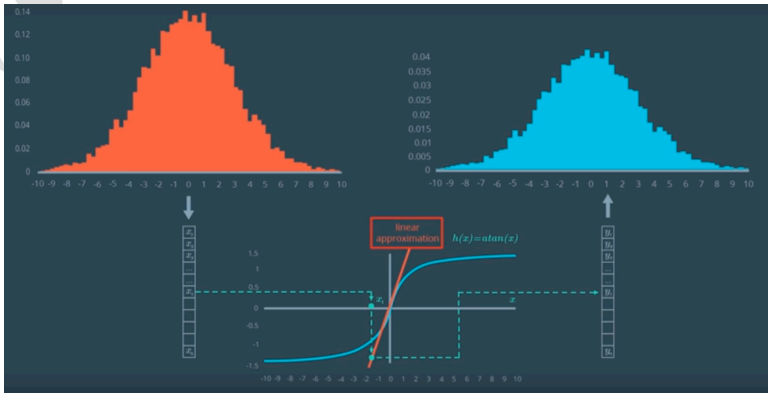
But the Measurement Vector from RADAR is $Z_t = \begin{bmatrix} \rho \\ \varphi \\ \dot{\rho} \end{bmatrix}$,

Now, Z has range, bearing and radial velocity, whereas X has linear position and velocity.
So, the measurement matrix M, needs to transform X from cartesian coordinates to polar coordinates. Thus, M cannot be a linear transformation. It will need to be computed manually.

- Since M is non-linear, we can no longer apply the standard Kalman filter equations.
- This is because, applying a non-linear function to a Gaussian does not result in a Gaussian.



One way to overcome this is to use a linear approximation of the function.
By using a linear approximation, we are essentially converting a non-linear function to a linear function and thus, we can work with the assumptions of a standard Kalman filter.
Below we see an example where using a linear approximation of the arctangent function, we are able to perform a linear transformation of the input Gaussian.



9.2.3.1 Linear Approximation of functions

We can use the general form of a Taylor series expansion to obtain a linear approximation of a non-linear function. The general form of a Taylor series expansion of a function f(x) at a point μ is given by:

$$f(x) \approx f(\mu) + \frac{\partial(f(\mu))}{\partial x}\,(x - \mu)$$

9.2.3.1.1 Example: Linear approximation of arctan function

$$h(x) = \arctan(x)$$

$$h`(x) = \frac{\partial(h(x))}{\partial x} = \frac{1}{1+x^2}$$

Thus, the linear approximation is:

$$f(x) \approx arctan(\mu) + \frac{1}{1+\mu^2}.(x - \mu)$$

Now, if we assume our predicted state density is defined as $N(0, \sigma^2)$, we get

$$f(x) \approx arctan(0) + \frac{1}{1+0}.(x - 0) = x$$

9.2.3.2 Multi-dimensional Linear approximation

The measurement matrix and measurement vector are multi-dimensional, so we will need to use the n-dimensional version of the Taylor series expansion.

The general form is as follows:

$$T(x) = f(a) + (x-a)^T \nabla_x f(a) + \frac{1}{2!}\,(x-a)^T \nabla_x^2 f(a)\,(x-a) + \cdots$$

Where,

∇_x is the Jacobian Matrix
∇_x^2 is the Hessian Matrix

Note: We will only use the Jacobian for our linear transformation assuming higher order terms are negligible.

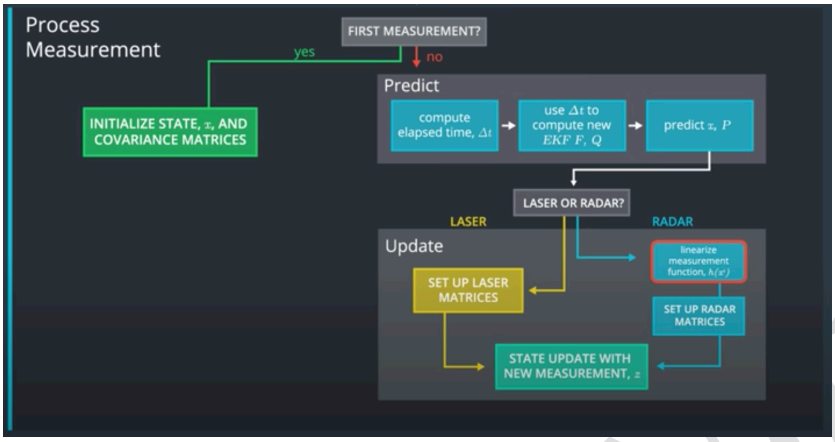
9.2.3.2.1 Multi dimension example

Let X = $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$, H = $\begin{bmatrix} a \\ b \end{bmatrix}$

$$Jacobian, \nabla_x H = \begin{bmatrix} \frac{\partial(a)}{\partial x_1} & \frac{\partial(a)}{\partial x_2} & \frac{\partial(a)}{\partial x_3} & \frac{\partial(a)}{\partial x_4} \\ \frac{\partial(b)}{\partial x_1} & \frac{\partial(b)}{\partial x_2} & \frac{\partial(b)}{\partial x_3} & \frac{\partial(b)}{\partial x_4} \end{bmatrix}$$

We can then use the above to compute the linear approximation. We will see this in more detail in the measurement step for RADAR.

9.2.4 RADAR: Measurement Update step



We are now in the measurement update phase for RADAR measurements.

Let the measurement be, $Z_t \sim N(M_t.X_t, \Sigma_{m_t})$

Where,

$M_t \rightarrow$ Measurement matrix

$\Sigma_{m_t} \rightarrow$ Measurement noise

This gives us:

$$\textbf{Measurement Vector}: Z_t = \begin{bmatrix} \rho \\ \varphi \\ \dot{\rho} \end{bmatrix},$$

From the prediction step we have,

$$X_t^- = \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}$$

We can compute the Jacobian to be:

$$Jacobian, \nabla_x Z = \begin{bmatrix} \frac{\partial(\rho)}{\partial p_x} & \frac{\partial(\rho)}{\partial p_y} & \frac{\partial(\rho)}{\partial v_x} & \frac{\partial(\rho)}{\partial v_y} \\ \frac{\partial(\varphi)}{\partial p_x} & \frac{\partial(\varphi)}{\partial p_y} & \frac{\partial(\varphi)}{\partial v_x} & \frac{\partial(\varphi)}{\partial v_y} \\ \frac{\partial(\dot{\rho})}{\partial p_x} & \frac{\partial(\dot{\rho})}{\partial p_y} & \frac{\partial(\dot{\rho})}{\partial v_x} & \frac{\partial(\dot{\rho})}{\partial v_y} \end{bmatrix}$$

On simplifying, we get

$$\begin{bmatrix} \frac{p_x}{\sqrt{p_x^2+p_y^2}} & \frac{p_y}{\sqrt{p_x^2+p_y^2}} & 0 & 0 \\ -\frac{p_y}{p_x^2+p_y^2} & \frac{p_x}{p_x^2+p_y^2} & 0 & 0 \\ \frac{p_y(v_x p_y - v_y p_x)}{(p_x^2+p_y^2)^{3/2}} & \frac{p_x(v_y p_x - v_x p_y)}{(p_x^2+p_y^2)^{3/2}} & \frac{p_x}{\sqrt{p_x^2+p_y^2}} & \frac{p_y}{\sqrt{p_x^2+p_y^2}} \end{bmatrix}$$

We use the above Jacobian as our measurement matrix.

Measurement Matrix, $M_t = \nabla_x Z$

For the radar measurements, we assume each measurement is affected by some noise. we assume it to be:

$$\rho \sim N(0, \Sigma_\rho)$$
$$\varphi \sim N(0, \Sigma_\varphi)$$
$$\dot{\rho} \sim N(0, \Sigma_{\dot{\rho}})$$

Also, we assume the measurement noise is a Gaussian with 0 mean and covariance Σ_{m_t}.

$$noise,\ \omega \sim N(0, \Sigma_{m_t})$$

This gives us:

Measurement Noise, $\Sigma_{m_t} = E[\omega\omega^T]$

$$= \begin{bmatrix} \Sigma_\rho & 0 & 0 \\ 0 & \Sigma_\varphi & 0 \\ 0 & 0 & \Sigma_{\dot{\rho}} \end{bmatrix}$$

9.2.4.1 Compute Measurement Update or correction

Inputs are the following:

- Predicted state in cartesian coordinates, X_t^-
- Prediction uncertainty, Σ_t^-
- Measurement vector in polar coordinates, Z_t
- Measurement noise, Σ_{m_t}

1. Convert predicted state to polar coordinates:

$$X_{tp}^- = \begin{bmatrix} \sqrt{p_x^2 + p_y^2} \\ \arctan\left(\frac{p_x}{p_y}\right) \\ \frac{p_x v_x + p_y v_y}{\sqrt{p_x^2 + p_y^2}} \end{bmatrix}$$

2. Compute measurement matrix, M_t

$$\begin{bmatrix} \frac{p_x}{\sqrt{p_x^2+p_y^2}} & \frac{p_y}{\sqrt{p_x^2+p_y^2}} & 0 & 0 \\ -\frac{p_y}{p_x^2+p_y^2} & \frac{p_x}{p_x^2+p_y^2} & 0 & 0 \\ \frac{p_y(v_x p_y - v_y p_x)}{(p_x^2+p_y^2)^{3/2}} & \frac{p_x(v_y p_x - v_x p_y)}{(p_x^2+p_y^2)^{3/2}} & \frac{p_x}{\sqrt{p_x^2+p_y^2}} & \frac{p_y}{\sqrt{p_x^2+p_y^2}} \end{bmatrix}$$

3. Compute Kalman Gain,

$$K_t = \Sigma_t^- \mathrm{M}_t^T \left(M_t \Sigma_t^- \mathrm{M}_t^T + \Sigma_{m_t}\right)^{-1}$$

4. Compute **updated measurement**

$$X_t^+ = X_t^- + K_t(Z_t - X_{tp}^-)$$

Note:

- We don't use M_t here since we already transformed X
- X_t^- is cartesian coordinates; X_{tp}^- is polar coordinates

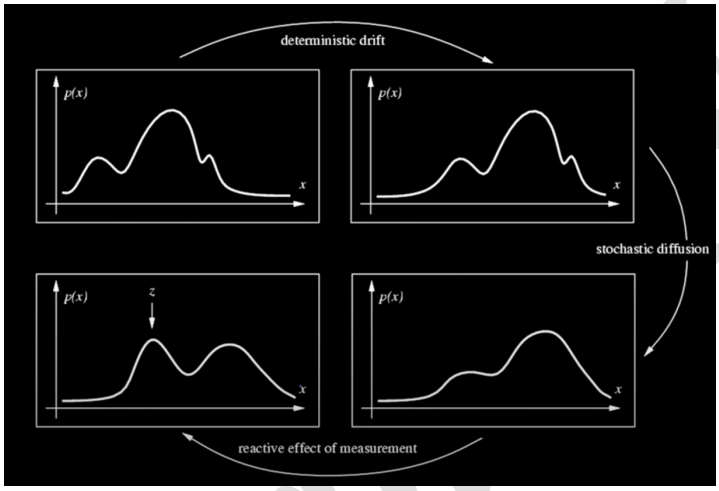
5. Compute **update uncertainty**

$$\Sigma_t^+ = (I - K_t M_t)\Sigma_t^-$$

10 Particle Filters

Kalman filter summary

- Method for tracking linear dynamical systems in Gaussian noise contexts
- Predicted\corrected state densities are Gaussian
    - We only need to maintain mean and covariance
    - Calculations are possible in closed form.
- Primary drawback is we assume a unimodal distribution (Gaussian)
- Realistically, we cannot assume Gaussians. It would be as follows:



10.1 Bayes Filter Framework

We will use z to represent measurements instead of y.
X represents the state.

We are **given** the following:

1. **Prior** Probability of system state: $p(x)$
2. **Action** (dynamical system) model: $p(x_t|\, u_{t-1}, x_{t-1})$
    a. This the probability or belief about state at time t, given
        i. u_{t-1} – is the perturbation or control. (control added to prior state)
        ii. x_{t-1} – state at previous time step.

3. **Sensor Model** (likelihood): $p(z|x)$
    a. This is *interpreted as* follows: If the object is really at x, what is the probability distribution of the measurement z.

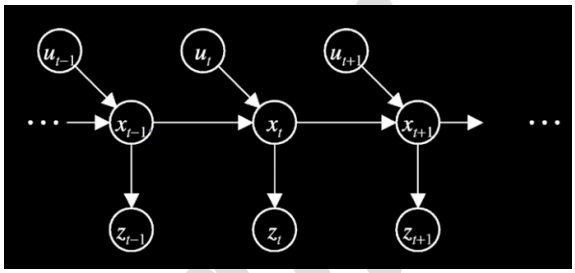
4. Stream of **observations z** and **action** (or control) data **u**:
    $data = \{u_1, z_2, \dots, u_{t-1}, z_t\}$

Goal

1. Estimate or predict the state X at time t.
2. The **posterior** (or belief) of the state is given by:
    $Bel(x_t) = P(x_t|\, u_1, z_2, \dots, u_{t-1}, z_t)$

Simplifying Assumptions

Similar to what we had in tracking as inference, we make some simplifying assumptions using the hidden Markov model:



- For the hidden state, only the immediate past and immediate control matter

    $P(x_t|\, x_{1:t-1}, z_{1:t-1}, u_{1:t}) \approx P(x_t|\, x_{t-1}, u_t)$
    (called the ***Dynamics model***)

- Measurements depend only on the current state.

    $P(z_t|\, x_{0:t}, z_{1:t-1}, u_{1:t}) \approx P(z_t|\, x_t)$
    (called the ***Observation Model, or sensor independence***)

Computing Belief

$$Bel(x_t) = P(x_t | u_1, z_2, \dots, u_{t-1}, z_t)$$

$$= \eta\, p(z_t \mid x_{0:t}, z_{1:t-1}, u_{1:t}).p(x_t | z_{2:t-1}, u_{1:t})$$

(Using Bayes Rule, η is the normalizer)

$$= \eta\, p(z_t \mid x_t).p(x_t | z_{2:t-1}, u_{1:t})$$

(Using sensor independence assumption)

$$= \eta\, p(z_t \mid x_t) \int p(x_t | z_{2:t-1}, u_{1:t}, x_{t-1}).p(x_{t-1} | z_{2:t-1}, u_{1:t})\, dx_{t-1}$$

(Conditioning on x_{t-1}, law of total probability)

$$= \eta\, p(z_t \mid x_t) \int p(x_t | x_{t-1}, u_t).p(x_{t-1} | z_{2:t-1}, u_{1:t})\, dx_{t-1}$$

(Using Markov assumption)

$$= \eta\, p(z_t \mid x_t) \int p(x_t | x_{t-1}, u_t).Bel(x_{t-1})\, dx_{t-1}$$

$$= \eta * sensorModel * prediction(x_t)$$

Likelihood

$$\text{Bel}(x_t) = \eta\, p(z_t \mid x_t) \int p(x_t | x_{t-1}, u_t).Bel(x_{t-1})\, dx_{t-1}$$

Prediction *before* taking measurements

10.2 Particle filters for tracking

In particle filters, instead of using an analytic representation of densities, we will use a set of weighted particles X. (assume, we have a map of the world from where particles can be drawn) Here, density is represented by both where the particles are and their weight. $p(x = x_0)$ is thus the probability of drawing an x with value close to x_0.

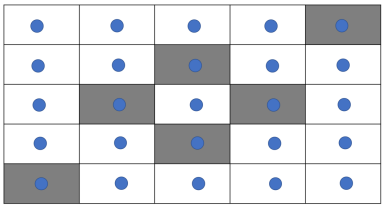
The goal is to get to a point as part of tracking where $p(x_t \in X_t) \approx p(x_t \mid z_{\{1:t\}})$ with equality when $n \to \infty$. i.e. the probability that we get a particle at a particular place (x_t) is the same as the probability of the state being at that place.

10.2.1.1 Example 1

We are given a map and some measurements. We need to find the final position of the robot.

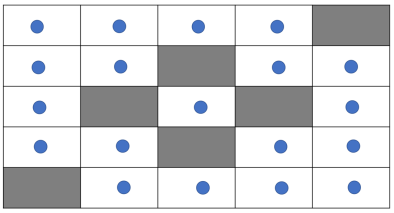
time	location	control
0	white	Up
1	white	Up
2	gray	Right
3	white	

At time 0, we have no idea about the position. So every square is equally likely. We have particles on every square.



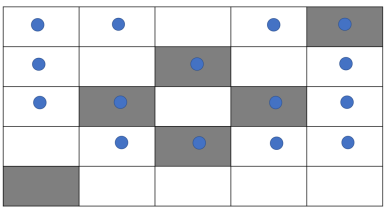
time	location	control
0	white	Up
1	white	Up
2	gray	Right
3	white	

We now a see a measurement of white. So particles in the grey squares are removed.



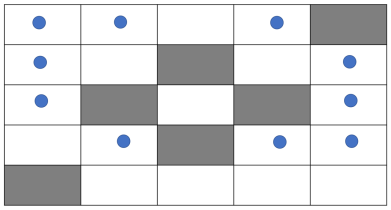
time	location	control
0	white	Up
1	white	Up
2	gray	Right
3	white	

Next we see an action of up. So all the remaining particles move up. Particles going outside the board are removed.

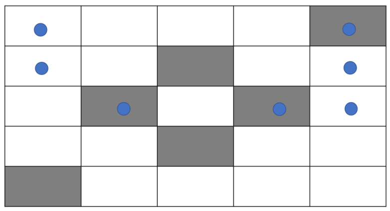


time	location	control
0	white	Up
1	white	Up
2	gray	Right
3	white	

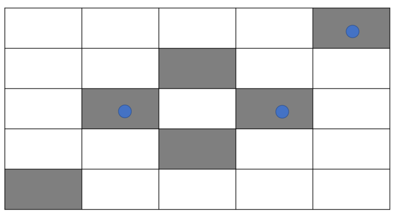
Following the above logic, we get the following sequence



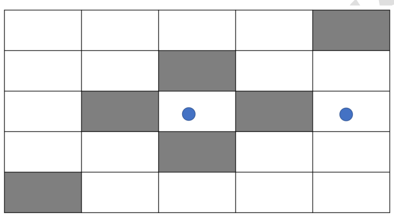
time	location	control
0	white	Up
1	white	Up
2	gray	Right
3	white	



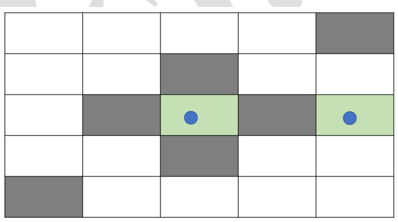
time	location	control
0	white	Up
1	white	Up
2	gray	Right
3	white	



time	location	control
0	white	Up
1	white	Up
2	gray	Right
3	white	



time	location	control
0	white	Up
1	white	Up
2	gray	Right
3	white	



time	location	control
0	white	Up
1	white	Up
2	gray	Right
3	white	

So now, we have two potential positions the robot could be at.

10.3 Particle Filter Algorithm

Assume we have a set of particles defined as follows:

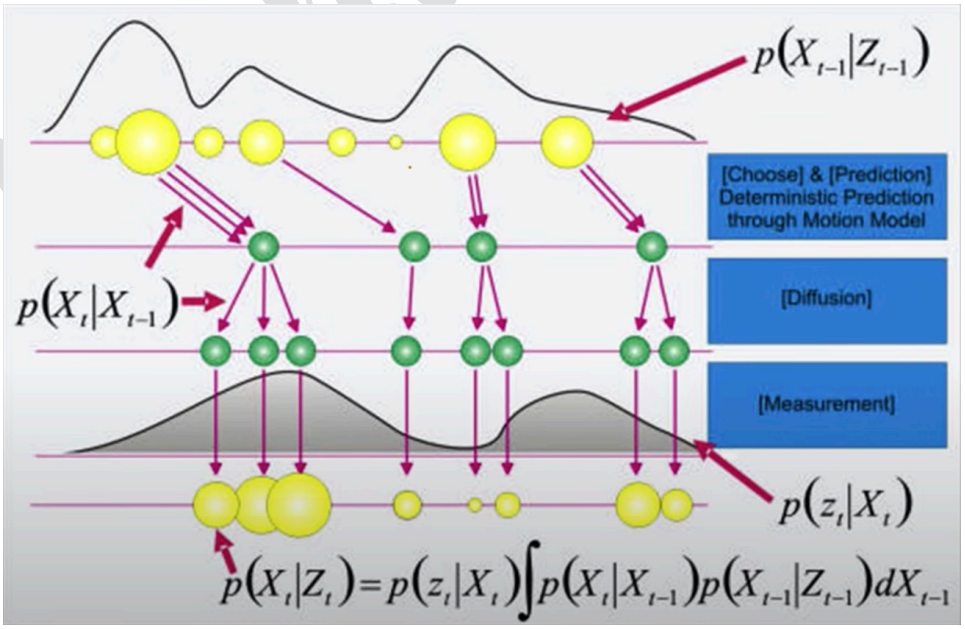
$$\{S_{t-1} = < x_{t-1}^{j}, w_{t-1}^{j} >, u_t, z_t\}$$

Where,

$$x_t \rightarrow Position\ of\ the\ particle\ at\ time\ t$$
$$w_t \rightarrow Weight\ of\ \ the\ \ particle\ at\ time\ t$$
$$u_t \rightarrow Action\ taken\ at\ time\ t$$
$$z_t \rightarrow Measurement\ at\ time\ t$$

1. Initial to an empty particle set, $S_t = \phi$, and $\eta = 0$
2. For $i = 1..n$ (generate i new samples)
3. Sample index $j(i)$ from the discrete distribution given by w_{t-1}
4. Sample x_t^i from $p(x_t|x_{t-1}, u_t)$ using $x_{t-1}^{j(i)}$ and u_t
5. $w_t^i = p(z_t|x_t^i)$ (reweight\compute weight)
6. $\eta = \eta + w_t^i$ (update normalization factor)
7. $S_t = S_t \cup \{< x_t^i, w_t^i >\}$ (add\update particle to set)
8. For $i = 1..n$
9. $w_t^i = \frac{w_t^i}{\eta}$ (Normalize weights)

10.3.1 Graphical Representation



10.4 Particle filters in reality

We need to define the following in reality:

1. What is the state x
2. What are the dynamics
3. What's a good sensor model

As a very simple model, we could have:

- State: Can be the location of an image patch (x,y)
- Dynamics: can be some random noise
- Sensor mode: mean squared difference of pixel intensities (i.e a similarity model. More similar implies more likely)

11 Additional References

- https://www.kalmanfilter.net/default.aspx
- Robust 3-D Motion Tracking from Stereo Images: A Model-less Method
- Vehicle Tracking and Motion Estimation Based on Stereo Vision Sequences
- Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting with a Single Convolutional Net
- VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection
- *Multiple Object Tracking using Kalman Filter and Optical Flow*
- *Kalman Filter Based Multiple Objects Detection-Tracking Algorithm Robust to Occlusion*
- *Tracking Multiple Moving Objects Using Unscented Kalman Filtering Techniques*
- *LIDAR-based 3D Object Perception*
- *Fast multiple objects detection and tracking fusing color camera and 3D LIDAR for intelligent vehicles*
- *3D-LIDAR Multi Object Tracking for Autonomous Driving*
- No Blind Spots: Full-Surround Multi-Object Tracking for Autonomous Vehicles using Cameras & LiDARs
- Multiple Sensor Fusion and Classification for Moving Object Detection and Tracking
- Robust 3-D Motion Tracking from Stereo Images: A Model-less Method
- Vehicle Tracking and Motion Estimation Based on Stereo Vision Sequences
- CONDENSATION - Conditional Density Propagation for Visual Tracking

End of Document