Filtering

Advanced Robotic Navigation

A brief review of Bayesian and Kalman Filtering

The theme of this course is navigation: we want to figure out where we are relative to other things. We refer to all possible places you might be as the *state space* and a specific sample of it is referred to as the *state* typically represented as x.

The state that is reported as the current location is referred to as a navigation solution.

The basic idea behind a probabilistic framework for navigation and the Bayes Filter is to represent your localization as a continuous probability distribution known as the *belief*. This straightforwardly translates to navigation solutions: you are currently located at the state with the highest belief.

The basic idea behind a probabilistic framework for navigation is to represent your localization as a continuous probability distribution known as the *belief*.

- This straightforwardly translates to navigation solutions: you are most likely located at the state with the highest belief.
- The foundation of this framework is Bayes' Theorem which allows you to describe how the belief about a random variable should change to account for the collected evidence (measurements).

$$p(x|z) = \frac{p(z|x)p(x)}{p(z)}$$

Bayes Filter

A Bayes' Filter calculates the probability of a state x_t at time t given all previous measurements and inputs $p(x_t|z_{1:t}, u_{1:t})$ in two steps.

Prediction step

- Propagate the belief to reflect updates in the motion of the system to predict the evolution of the state
- Takes the prior belief about the state and control input
- Calculates the predicted belief about the state from the process or motion model

Measurement (update) step

- Measure or sense the world and update the belief to reflect these external inputs
- Takes the predicted belief and a measurement
- Calculates the posterior belief about the state according to the measurement model

A Bayes' Filter is recursive and calculates the probability accounting for *all* prior measurements and controls.

Mathematically impossible to implement for any significant duration of time.

Simplify the system by using assuming it is a first-order Markov chain:

- Assume each state is conditionally independent
- Only the previous state and current controls effects the current state

Simplification permits recursion via sequential processing making the filter tractable.

A Bayes filter is only two functions: predict and update

for (u, z) in (controls, measurements):

    bel(x) = predict(x, u) * bel(x)

    bel(x) = update(x, z) * bel(x)

Consider a one-dimensional constant velocity system where the state is position, controls are velocity, and the measurement is position. The sensor is modeled as a normal distribution with accuracy σ.

predict(state, u=(control, dt)):

    return p(state + control * dt)

update(state, measurement):

    return (1 / ($\sigma \sqrt{2\pi}$)) * exp(-0.5 ((state – measurement)/σ)^2)

The Problems with Bayes' Filter

While potential useful in some small-scale scenarios, a Bayes Filter is rarely used in practice for several reasons. To start, the Bayes Filter calculates the belief across the *entire* state space. For practical navigation purposes the problem is intractable. In these scenarios, we typically deal with two or three position states and their associated velocities and angular velocities.

The Bayes Filter calculates the belief across the *entire* state space and for practical navigation purposes, while calculable, the problem is intractable.

- Three position states, velocities, angular velocities…
- Floating point probabilities

Take a typical soccer field (100m x 75m) represented by a binary occupancy grid

- At $1\ \mathrm{m}^2$ resolution, 7,500 position states and 56,250,000
- Single precision (32 bit) floats: 225 MB of memory just for positions!

A fairly basic two-dimensional navigation system charged with positioning a simple differential drive robot on a flat warehouse floor is already a five-state problem at a minimum (x-, y-position, orientation, body-axis velocity, angular velocity). Orientation is fairly limited (0 to 360 degrees), and we could reasonably bound the velocities based on the performance characteristics of the robot, but now imagine you're working in a typical distribution center with a warehouse size of 50,000 to 100,000 m^2. Even if you were to use a discrete Bayes Filter and only calculate position down to 1 meter, with a 100,000 m^2 warehouse, the first order states alone already presents 36 million individual state probabilities to calculate.

Using 32-bit single-precision floating point variables, that would require around 9.2 GB of memory just to hold the discrete probability distribution over three states. This will obviously grow when we add in the velocities. Suffice it to say, even with great computing hardware, the time it would take to compute the posterior belief over such a state space would take too long to be useful with which to navigate.

This offers up some motivation as to how we can improve the Bayes Filter. We *like* the probabilistic technique it presents: it allows us to derive certain estimates from uncertain measurements, but we need to find a way to make it practical.

Recall some classical statistics: because we model the navigation state as a collection of *independent* random variables, the central limit theorem states that if we measure our state enough times the samples will resemble a normal distribution. Alternatively, we can make the design choice to *parameterize* our state space as an estimate (mean) and uncertainty (covariance).

Simplifying assumptions

How do we solve the problems with Bayes' Filter?

Recall some classical statistics:

- Because we model the navigation state as a collection of *independent* random variables, the central limit theorem states that if we measure our state enough times the samples will resemble a normal distribution
- When modeling something probabilistically, a normal distribution is a reasonable model.
- Alternatively, we can make the design choice to *parameterize* our state space as an estimate (mean) and uncertainty (covariance).
- Linear transformations of normal distributions remain linear

With this motivation to parameterize as a normal distribution and simply the state space we can (must) make two assumptions:

The prior state is represented as a normal distribution: $p(x_0) \sim N(\mu_o, \Sigma_0)$

The process model and measurement model are *linear* with additive *normal* white noise

$$x_t = A_t x_{t-1} + B_t u_t + n_t$$

$$z_t = C_t x_t + v_t$$

The Kalman Filter

A Kalman Filter is a tractable implementation of a Bayesian Filter that satisfies these assumptions.

Instead of tracking the belief across the entire state space, it models the belief as a normal distribution and thus only needs to track the mean and variance/covariance.

Reduces the memory load on the computer and is easily implemented in five lines of linear algebra (highly optimized libraries).

Full derivation available in *Probabilistic Robotics* Chapter 3.2, useful to look over and refresh.

Prediction is relatively straightforward, prior probability of state x_{t-1} is given by the normal distribution parameterized by the prior mean and covariance: $p(x_{t-1}) \sim \mathcal{N}(\mu_{t-1}, \Sigma_{t-1})$

- State transition model: $f(x_t|x_{t-1},\ u_t) = A_t x_{t-1} + B_t u_t + \mathcal{N}(0, Q_t)$

Parameterize and apply expected value and covariance operations:

- $\bar{\mu}_t = A\mu_{t-1} + Bu_t$
- $\bar{\Sigma}_t = A\Sigma_{t-1}A^T + Q$

The update of μ_t conditioned on z_t is normal:

- $\mu_{x_t|z_t} = \bar{\mu}_t + \bar{\Sigma}_t C^T (C\bar{\Sigma}_t C^T + R)^{-1}(z_t - C\bar{\mu}_t)$
- $\Sigma_{x_t|z_t} = \bar{\Sigma}_t - \bar{\Sigma}_t C^T (C\bar{\Sigma}_t C^T + R)^{-1} C\bar{\Sigma}_t$

For notation's sake we'll define the matrix K as the Kalman gain:

- $K = \bar{\Sigma}_t C^T (C\bar{\Sigma}_t C^T + R)^{-1}$
- Basically: a matrix of weightings for how much to trust the sensor against the prediction

Update step:

- $K_t = \bar{\Sigma}_t C^T (C\bar{\Sigma}_t C^T + R)^{-1}$
- $\mu_t = \bar{\mu}_t + K_t(z_t - C\bar{\mu}_t)$
- $\Sigma_t = \bar{\Sigma}_t - K_t C\bar{\Sigma}_t$

The Kalman Gain for a perfect sensor

$$K_t = \bar{\Sigma}_t C^T (C\bar{\Sigma}_t C^T + R)^{-1} \to C^{-1}$$

$$\mu_t = \bar{\mu}_t + K_t(z_t - C\bar{\mu}_t) \to C^{-1} z_t$$

$$\Sigma_t = \bar{\Sigma}_t - K_t C\bar{\Sigma}_t = \bar{\Sigma}_t - C^{-1} C\bar{\Sigma}_t \to 0$$

The Kalman Gain for a bad (or no) sensor:

$$K_t = \bar{\Sigma}_t C^T (C\bar{\Sigma}_t C^T + R)^{-1} \to 0$$

$$\mu_t = \bar{\mu}_t + K_t(z_t - C\bar{\mu}_t) \to \bar{\mu}_t$$

$$\Sigma_t = \bar{\Sigma_t} - K_t C \bar{\Sigma_t} = \bar{\Sigma_t} - 0\bar{\Sigma_t} = \bar{\Sigma_t}$$

An example

Simple case: 2D planer motion with two degrees of freedom, no control input, acceleration modeled as noise

- $\boldsymbol{x} = (p_x,\ p_y, v_x,\ v_y)$
- $\boldsymbol{z} = (z_x,\ z_y)$

Continuous time system is clearly linear:

- $\dot{x}_t = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}_t x_t + \eta_t$
- $F = (I + A\delta t) = \begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_t$
- This follows our intuition of basic kinematics: $\bar{x} = x + vt$

No control inputs modeled, so B matrix is zero.

Acceleration is modeled as noise so E matrix is one on the diagonals for the velocity terms:

- $E = \begin{bmatrix} \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & I_2 \end{bmatrix}$
- $V = E\delta t = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \delta t & 0 \\ 0 & 0 & 0 & \delta t \end{bmatrix}$

Measurements are of positions only, so measurement matrix is straightforward:

- $C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$

Kalman Filter prediction equations:

- $\bar{\mu}_t = \overline{\begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}_t} = \begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_t \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}_{t-1}$
- $\bar{\Sigma}_t = \begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_t \Sigma_t \begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_t^T + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \delta t & 0 \\ 0 & 0 & 0 & \delta t \end{bmatrix} Q \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \delta t & 0 \\ 0 & 0 & 0 & \delta t \end{bmatrix}^T$

Kalman Filter update equations

- $K = \bar{\Sigma}_t \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}^T \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \bar{\Sigma}_t \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}^T + R \right)^{-1}$
- $\mu_t = \bar{\mu}_t + K_t \left(z_t - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \bar{\mu}_t \right)$
- $\Sigma_t = \bar{\Sigma}_t - K_t \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \bar{\Sigma}_t$

This is a basic example of two-dimensional target tracking

- Can easily be extend to three dimensions
- Alternatively, can be used for navigation when you have an external reference for position and orientation

Nonlinearities

What do we assume with a Kalman Filter?

- The prior state is best represented by a normal distribution parameterized by a mean and covariance.
- The process model and measurement model are linear with additive white (normally distributed) noise.

What are the problems with this assumption?

- Dynamic systems (and most things in life) are very rarely completely linear.
- How do we account for nonlinear processes and measurements while preserving our linear normally distributed estimate?

Recall our previous two-dimensional example: 2D, planer, 2 DOF.

- Able to use the Kalman Filter due to holonomic motion example and external position measurements
- In practice, this would require knowledge of orientation or a specific holonomic drive
- That model was more applicable to target tracking rather than localization/navigation

Now consider a localization or navigation problem in a two-dimensional planer space with *three* degrees of freedom (x, y, θ).

How can we derive a global position by only measuring from the body frame?

In the body frame, what can we measure?

- Body frame accelerations and velocities (IMU, odometry): v, ω
- Heading (odometry, magnetic compass): θ
- Global position (GPS): p_x, p_y

Process Model

System propagation equation transforming body frame velocities into global frame position:

$$\begin{bmatrix} p_x \\ p_y \\ \theta \\ v \\ \omega \end{bmatrix}_t = \begin{bmatrix} 1 & 0 & 0 & t\cos\theta & 0 \\ 0 & 1 & 0 & t\sin\theta & 0 \\ 0 & 0 & 1 & 0 & t \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ \theta \\ v \\ \omega \end{bmatrix}_{t-1} + B_t u_t + \eta_t$$

Non-linear propagation method is simple to resolve (basic 2D kinematics), simply update the matrix every iteration as a function of the current state: $F = F(\mu_t,\ t)$

GPS provides a convenient way to measure global position from the body frame: $z_t = \begin{bmatrix} p_{x,z} \\ p_{y,z} \end{bmatrix}_t$

Full system (covariance update omitted)

$$\mu_t = \begin{bmatrix} 1 & 0 & 0 & t\cos\theta & 0 \\ 0 & 1 & 0 & t\sin\theta & 0 \\ 0 & 0 & 1 & 0 & t \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ \theta \\ v \\ \omega \end{bmatrix}_{t-1} + B_t u_t + \eta_t + K_t \left(\begin{bmatrix} p_{x,z} \\ p_{y,z} \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \bar{\mu}_t \right)$$

First problem: nonlinear propagation

- Trigonometric functions in the propagation matrix introduce nonlinearities
- Remember affine transformation properties: a normal distribution put through a nonlinear function results in a non-normal distribution!

Measurement Model

What if a GPS is not available?

- How do we measure a global position $(p_{x,z}, p_{y,z})$ in the body frame?
- We can measure range and bearing to a known position (z_x, z_y)
- Can we produce a matrix that relates the current state to a range and bearing?
- $\begin{bmatrix} l \\ \phi \end{bmatrix} = [?]x_t + \nu_t$
- $[?] = \begin{bmatrix} \sqrt{(z_x - p_x)^2 + (z_y - p_y)^2} \\ \operatorname{atan}\frac{z_y - p_y}{z_x - p_x} - \theta \end{bmatrix}$
- This measurement matrix cannot be applied to the linear model above

The nonlinearities in this system introduce two significant problems

- Sending any normally distributed state variable through nonlinear function violates the normally distributed assumption

- If we cannot directly measure global position, the measurement update must be a function of the current state and violates the linear system model

One of the primary nonlinearities we're concerned about with navigation comes from IMUs as inertial outputs frequently form the basis of modern navigation systems.

Nonlinearities in Inertial Navigation

One of the biggest issues with nonlinearities comes from using the odometry provided by inertial measurement units.

Recall the point of IMUs, provides the ability to sense accelerations and angular velocities allowing for integration up to positions and orientations

Two primary nonlinearities and concerns:

1. Uncompensated, IMUs will detect and report gravitational accelerations (~9.81m/s^2) so need to detect orientation to properly filter out
2. The trigonometry functions used to describe and decompose the orientation to the cartesian axes are nonlinear

The typical 15-state strapdown INS state vector

- $\mathbf{x} = \begin{bmatrix} p_x & p_y & p_z & \phi & \theta & \psi & v_x & v_y & v_z & b_{g_x} & b_{g_y} & b_{g_z} & b_{a_x} & b_{a_y} & b_{a_z} \end{bmatrix}^T = \begin{bmatrix} \boldsymbol{p} & \boldsymbol{q} & \dot{\boldsymbol{p}} & \boldsymbol{b}_g & \boldsymbol{b}_a \end{bmatrix}^T$

From IMU output:

- $\boldsymbol{\omega}(t) = \left(\boldsymbol{I}_3 + \mathbf{M}_g\right)^{-1}(\bar{\boldsymbol{\omega}} - \mathbf{b}_g - \eta_g)$
- $\mathbf{a}(t) = (\boldsymbol{I}_3 + \mathbf{M}_a)^{-1}(\bar{\boldsymbol{f}} - \mathbf{b}_a - \eta_a) - R(\mathbf{q})\mathbf{g}$

Integrate up IMU accelerations and angular velocities to get velocities, positions, and orientations.

- $v(t) = v(t_0) + \int \boldsymbol{a}(t)dt = v(t_0) + \int \left((\boldsymbol{I}_3 + \mathbf{M}_a)^{-1}(\bar{\boldsymbol{f}} - \mathbf{b}_a - \eta_a) - R_i(\mathbf{q})\mathbf{g}\right)dt$
- $p(t) = p(t_0) + \int \boldsymbol{v}(t)dt = p(t_0) + v(t_0)dt + \int \left(\int \left((\boldsymbol{I}_3 + \mathbf{M}_a)^{-1}(\bar{\boldsymbol{f}} - \mathbf{b}_a - \eta_a) - R_i(\mathbf{q})\mathbf{g}\right)dt\right) dt$
- $q(t) = q(t_0) + \int \boldsymbol{\omega}(t)dt = q(t_0) + \int \left(\boldsymbol{I}_3 + \mathbf{M}_g\right)^{-1}\left(\bar{\boldsymbol{\omega}} - \mathbf{b}_g - \eta_g\right)dt$

Continuous-time process model: $\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{p}} \\ G(\mathbf{q})^{-1}\mathbf{u}_\omega \\ \mathbf{g} + R(\mathbf{q})\mathbf{u}_a \\ \mathbf{b}_g \\ \mathbf{b}_a \end{bmatrix}$

Primary concerns are the rotation matrices $G(\mathbf{q})$ and $R(\mathbf{q})$ as trigonometric functions are nonlinear.

Linearization

Let's relax the strict linear model assumption and go back to Bayes and a continuous system:

- Prior: $p(x_0) \sim \mathcal{N}(\mu_0, \Sigma_0)$
- Transition model: $\dot{x} = f(x, u, \eta)$
- Measurement model: $z = g(x, v)$

Prediction step

- Process model is nonlinear, and we need to convert continuous time dynamics to discrete time
- We will consider a finite time interval $\tau = [t', t)$ where $\delta t = t - t'$, $\bar{t} = \delta t \to 0$
- Options to solve:
  - Classical integration of $f(x, u, \eta)$ over τ: $x_{\bar{t}} = \Phi(\bar{t}, x_{t-1}, u, \eta)$, difficult to do
  - Numerical one-step Euler integration, easier to do, but approximate and requires linearization

Linearizing the Process Model

Linearize the dynamics about $x = \mu_{t-1}, u = u_t, \eta = 0$

$$\dot{x} \approx f(\mu_{t-1}, u_t, 0) + \frac{\delta f}{\delta x}\Big|_{\mu_{t-1}, u_t, 0}(x - \mu_{t-1}) + \frac{\delta f}{\delta u}\Big|_{\mu_{t-1}, u_t, 0}(u - u_t) + \frac{\delta f}{\delta \eta}\Big|_{\mu_{t-1}, u_t, 0}(\eta - 0)$$

Some substitutions:

- $A_t = \frac{\delta f}{\delta x}\Big|_{\mu_{t-1}, u_t, 0}$
- $B_t = \frac{\delta f}{\delta u}\Big|_{\mu_{t-1}, u_t, 0}$
- $U_t = \frac{\delta f}{\delta \eta}\Big|_{\mu_{t-1}, u_t, 0}$

Dynamics are now linear with respect to the state:

$$\dot{x} \approx f(\mu_{t-1}, u_t, 0) + A_t(x - \mu_{t-1}) + B_t(u - u_t) + U_t(\eta - 0)$$

One-step integration:

$$\bar{x}_t \approx x_{t-1} + f(x_{t-1}, u_t, \eta_t)\delta t$$

$$\approx x_{t-1} + \delta t\,(f(\mu_{t-1}, u_t, 0) + A_t(x - \mu_{t-1}) + B_t(u_t - u_t) + U_t\eta_t)$$

$$\approx (I + \delta t\, A_t)x_{t-1} + \delta t(f(\mu_{t-1}, u_t, 0) - A_t\mu_{t-1}) + \delta t\, U_t\eta_t$$

$$\approx F_t x_{t-1} + \delta t\,(f(\mu_{t-1}, u_t, 0) - A_t\mu_{t-1}) + V_t\eta_t$$

We now have some components that we can work with that look like the Kalman Filter.

- $\bar{\mu}_t = \mu_{t-1} + \delta t\, f(\mu_{t-1},\, u_t,\, 0)$
- $\bar{\Sigma}_t = F_t \Sigma_{t-1} F_t^T + V_t Q_t V_t^T$
- $F_t = I + \delta t\, A_t$

Linearizing the Measurement Model

Now we need to do the same for the observation model:

$$g(x,\, v) \approx g(\bar{\mu}_t,\, 0) + \left.\frac{\delta g}{\delta x}\right|_{\bar{\mu}_t, 0} (x - \bar{\mu}_t) + \left.\frac{\delta g}{\delta v}\right|_{\bar{\mu}_t, 0} (v - 0)$$

Again, substituting:

- $G_t = \left.\frac{\delta g}{\delta x}\right|_{\bar{\mu}_t, 0}$
- $W_t = \left.\frac{\delta g}{\delta v}\right|_{\bar{\mu}_t, 0}$

This again results in a linear model with respect to the state:

$$z_t = g(x_t,\, v_t) \approx g(\bar{\mu}_t, 0) + G_t(x_t - \bar{\mu}_t) + W_t v_t$$

Putting these two linearized models together is the basis for the Extended Kalman Filter (EKF) which uses these linearized models to fit the system into a Kalman Filter framework.

The Extended Kalman Filter

Very similar to the base linear Kalman Filter:

- Prediction:
  - $\bar{\mu}_t = \mu_{t-1} + \delta t\, f(\mu_{t-1},\, u_t,\, 0)$
  - $\bar{\Sigma}_t = F_t \Sigma_{t-1} F_t^T + V_t Q_t V_t^T$
- Measurement update:
  - $K = \bar{\Sigma}_t G_t^T (G_t \bar{\Sigma}_t G_t^T + W_t R_t W_t^T)^{-1}$
  - $\mu_t = \bar{\mu}_t + K_t\big(z_t - g(\bar{\mu}_t,\, 0)\big)$
  - $\Sigma_t = \bar{\Sigma}_t - K_t G_t \bar{\Sigma}_t$

Note that the only real changes are on the noise terms (Q_t and R_t), and how the current state maps to the measurement

Matrices are calculated via Taylor series expansion of $f(x_{t-1},\, u_t,\, \eta_t)$ and $g(x_t,\, v_t)$ at each time interval to calculate the F_t and V_t, and G_t and W_t matrices, respectively.

- $F_t = I + \delta t \left.\frac{\delta f}{\delta x_t}\right|_{\bar{\mu}_t,\, u_t,\, 0}$

- $V_t = \delta t \left.\frac{\delta f}{\delta \eta_t}\right|_{\bar{\mu}_t,\ u_t,\ 0}$
- $G_t = \left.\frac{\delta g}{\delta x_t}\right|_{\bar{\mu}_t\ ,\ 0}$
- $W_t = \left.\frac{\delta g}{\delta v_t}\right|_{\bar{\mu}_t\ ,\ 0}$

This describes a first order Taylor series approximation EKF

- Higher order EKFs use additional terms from the Taylor series to approximate F_t and G_t
- Only beneficial with lower measurement noise

Try not to get too hung up on specific notation, understand what each variable and function is saying

- Some authors frequently use A_t, and F_t interchangeably
- Measurement function and matrix sometimes use h and H_t

Let's return to our previous example:

- 2D planer motion, 3 degrees of freedom: $x_t = [p_x,\ p_y,\ \theta,\ v,\ \omega]$
- Body frame range and bearing measurement to a known landmark: $z_t = [r,\ \phi]$
- No control inputs, noise is only on the velocity terms.

$$\dot{x}_t = f(x_t,\ u_t,\ \eta_t) = \begin{bmatrix} 0 & 0 & 0 & t\cos\theta & 0 \\ 0 & 0 & 0 & t\sin\theta & 0 \\ 0 & 0 & 0 & 0 & t \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ \theta \\ v \\ \omega \end{bmatrix}_{t-1} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathcal{N}(0,\ Q_t) = \begin{bmatrix} vt\cos\theta \\ vt\sin\theta \\ \omega t \\ \eta_v \\ \eta_\omega \end{bmatrix}$$

- Measurement returns range and bearing to a known landmark with position $(z_x,\ z_y)$

$$g(\bar{\mu}_t, \nu_t) = \begin{bmatrix} r \\ \phi \end{bmatrix} = \begin{bmatrix} \sqrt{(z_x - \bar{p}_x)^2 + \left(z_y - \bar{p}_y\right)^2} + \nu_{t,r} \\ \operatorname{atan}\dfrac{z_y - \bar{p}_y}{z_x - \bar{p}_x} - \theta + \nu_{t,\phi} \end{bmatrix}$$

Linearization:

$$\dot{x} \approx f(\mu_{t-1}, u_t, 0) + (x - \mu_{t-1})\left.\frac{\delta f}{\delta x}\right|_{\mu_{t-1},\, u_t, 0} + (u - u_t)\left.\frac{\delta f}{\delta u}\right|_{\mu_{t-1},\, u_t, 0} + (\eta - 0)\left.\frac{\delta f}{\delta \eta}\right|_{\mu_{t-1},\, u_t, 0}$$

$$A_t = \frac{\delta f}{\delta x}\Big|_{\mu_{t-1},\, u_t, 0} = \begin{bmatrix} \frac{\delta p_x}{\delta p_x} & \frac{\delta p_x}{\delta p_y} & \frac{\delta p_x}{\delta \theta} & \frac{\delta p_x}{\delta v} & \frac{\delta p_x}{\delta \omega} \\ \frac{\delta p_y}{\delta p_x} & \frac{\delta p_y}{\delta p_y} & \frac{\delta p_y}{\delta \theta} & \frac{\delta p_y}{\delta v} & \frac{\delta p_y}{\delta \omega} \\ \frac{\delta \theta}{\delta p_x} & \frac{\delta \theta}{\delta p_y} & \frac{\delta \theta}{\delta \theta} & \frac{\delta \theta}{\delta v} & \frac{\delta \theta}{\delta \omega} \\ \frac{\delta v}{\delta p_x} & \frac{\delta v}{\delta p_y} & \frac{\delta v}{\delta \theta} & \frac{\delta v}{\delta v} & \frac{\delta v}{\delta \omega} \\ \frac{\delta \omega}{\delta p_x} & \frac{\delta \omega}{\delta p_y} & \frac{\delta \omega}{\delta \theta} & \frac{\delta \omega}{\delta v} & \frac{\delta \omega}{\delta \omega} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -v_x \sin\theta & \cos\theta & 0 \\ 0 & 0 & v_y \cos\theta & \sin\theta & 0 \\ 0 & 0 & 0 & 0 & t \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B_t = \frac{\delta f}{\delta u}\Big|_{\mu_{t-1},\, u_t, 0} = [0]$$

$$U_t = \frac{\delta f}{\delta \eta}\Big|_{\mu_{t-1},\, u_t, 0} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

One-step Euler integration:

$$F_t = I + \delta t\, \frac{\delta f}{\delta x_t}\Big|_{\bar{\mu}_t,\, u_t,\, 0} = \begin{bmatrix} 1 & 0 & -v\, \delta t \sin\theta & \cos\theta & 0 \\ 0 & 1 & v\, \delta t \cos\theta & \sin\theta & 0 \\ 0 & 0 & 1 & 0 & \delta t^2 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$V_t = \delta t\, \frac{\delta f}{\delta \eta_t}\Big|_{\bar{\mu}_t,\, u_t,\, 0} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \delta t & 0 \\ 0 & 0 & 0 & 0 & \delta t \end{bmatrix}$$

Now the measurement matrices:

$$g(\bar{\mu}_t, \nu_t) = \begin{bmatrix} r \\ \phi \end{bmatrix} = \begin{bmatrix} \sqrt{(z_x - \bar{p}_x)^2 + \left(z_y - \bar{p}_y\right)^2} + \nu_{t,r} \\ \operatorname{atan} \dfrac{z_y - \bar{p}_y}{z_x - \bar{p}_x} - \theta + \nu_{t,\phi} \end{bmatrix}$$

$$g(x,\, \nu) \approx g(\bar{\mu}_t,\, 0) + (x - \bar{\mu}_t) \frac{\delta g}{\delta x}\Big|_{\bar{\mu}_t, 0} + (\nu - 0) \frac{\delta g}{\delta \nu}\Big|_{\bar{\mu}_t, 0}$$

Summary of matrices:

$$F_t = \begin{bmatrix} 1 & 0 & -v\, \delta t \sin\theta & \cos\theta & 0 \\ 0 & 1 & v\, \delta t \cos\theta & \sin\theta & 0 \\ 0 & 0 & 1 & 0 & t \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad B_t = [0]$$

$$U_t = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad Q_t = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \eta_v & 0 \\ 0 & 0 & 0 & 0 & \eta_\omega \end{bmatrix}$$

$$V_t = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \delta t & 0 \\ 0 & 0 & 0 & 0 & \delta t \end{bmatrix}, \qquad \mathrm{R_t} = \begin{bmatrix} \nu_{t,r} & 0 \\ 0 & \nu_{t,\phi} \end{bmatrix}$$

$$G_t = \begin{bmatrix} -\frac{z_x - p_x}{\sqrt{(z_x-p_x)^2+(z_y+p_y)^2}} & -\frac{z_y - p_y}{\sqrt{(z_x-p_x)^2+(z_y+p_y)^2}} & 0 & 0 & 0 \\ \frac{z_y - p_y}{(z_x-p_x)^2+(z_y+p_y)^2} & \frac{p_x - z_x}{(z_x-p_x)^2+(z_y+p_y)^2} & -1 & 0 & 0 \end{bmatrix}$$

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

F_t and G_t need to be calculated at each iteration. If your system's noise parameters are constant, then U_t, Q_t, W_t, and R_t are constants.

Assumptions

- Normal distribution assumption is relaxed, but still relies on it being a reasonably accurate approximation
- Differentiability of the system model and measurement model (at least first order)
- Small perturbations

Strengths

- Simple, has the strengths of the Kalman Filter with added flexibility
- Works with generic models

Limitations and weaknesses

- Limited to systems with lower-order nonlinearities
- Requires repeated calculation of Jacobian (partial derivatives) matrices, more problematic the higher dimensions you have
- Still limited to unimodal distributions

So, we can now do Kalman Filtering on non-linear systems, but we added to the computational complexity and still can't handle higher order nonlinearities very well. That kind of stinks...

The Unscented Kalman Filter

We start from the same point as the EKF.

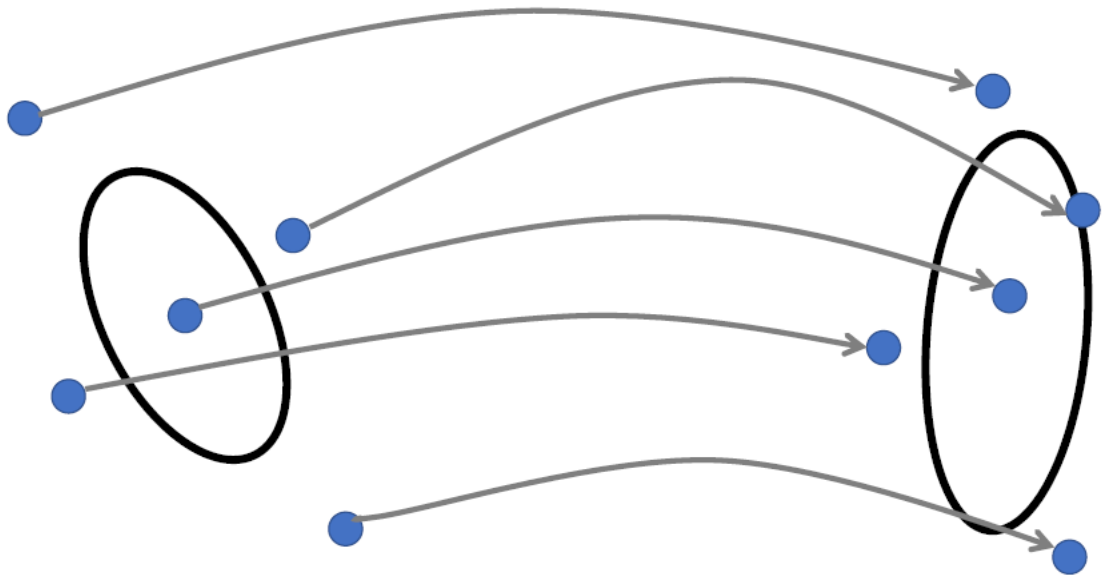
- Prior: $p(x_0) \sim \mathcal{N}(\mu_0, \Sigma_0)$
- Continuous time generic (nonlinear) process model with normal white noise (η): $\dot{x} = f(x, u, \eta)$
- Observation model is a generic (nonlinear) function with normal white noise (v): $z = g(x, v)$

We now include something called the unscented transformation.

- This is an alternative linearization approach to the Taylor series used in the EKF
- Performs a stochastic linearization using a weighted statistical linear regression
- Essentially, picks a representative number of points, propagates them, and attempts to reconstruct a normal distribution after propagation

Named 'unscented' because the authors thought the EKF stunk.

The Unscented Transform



Input

Normal distribution parameters

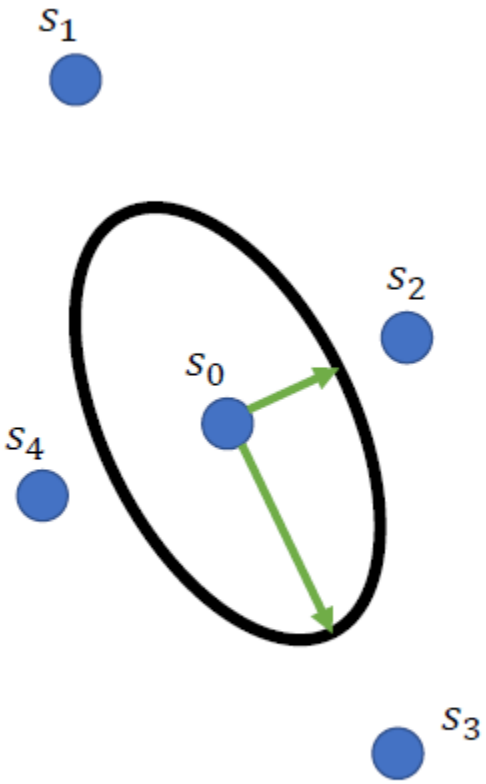
- Mean
- Covariance
- Dimension of the system or size of the state vector (d)

Output

Sigma points

- Discrete selections of probability distribution
- Number of points: $2d + 1$

- Each point is weighted which corresponds to how they are used to reconstruct the distribution
- The sigma points are a column-wise array corresponding to:
- $\mathcal{X}_t = [\mu_t,\ \mu_t \pm \sqrt{(d+\lambda)\Sigma_t}]$
- Alternatively, as a list:
- $\mathcal{X}_t^{[0]} = \mu_t$
- $\mathcal{X}_t^{[i]} = \mu + \sqrt{(d+\lambda)\Sigma_t}$ for $i = 1, \dots, d$
- $\mathcal{X}_t^{[i]} = \mu - \sqrt{(d+\lambda)\Sigma_t}$ for $i = d+1, \dots, 2d$
- Where λ is a scaling parameter corresponding to the scaling parameters α and κ:
- $\lambda = \alpha^2(d+\kappa) - d$



Each sigma point has a corresponding weight where w_c is used to reconstruct the covariance and w_m is used to calculate the mean:

$$w_m^{[0]} = \frac{\lambda}{d+\lambda}$$

$$w_c^{[0]} = w_m^{[0]} + 1 - \alpha^2 + \beta$$

$$w_m^{[i]} = w_c^{[i]} = \frac{1}{2(d+\lambda)} \text{ for } i = 1, \dots, 2d$$

α and κ

- Control the spread of the points
- Values are dependent on the scenario being modeled
- Typical choice is $\alpha = 10^{-3}$ and $\kappa = 1$

$$\beta$$

- Factor relating to the distribution
- If the distribution is truly normal, $\beta = 2$ is optimal
- Reinforces the weight of the sigma point corresponding to the mean

The UKF

With the unscented transform used to linearize, the system the becomes relatively simple despite the somewhat complex notation.

- Prediction step
  - Calculate sigma points from prior estimate $(\mu_{t-1}, \Sigma_{t-1})$
  - Propagate mean and covariance using a weighted sum $(\bar{\mu}_t, \bar{\Sigma}_t)$
- Update step
  - Calculate propagated sigma points from propagated mean and covariance
  - Transform the sigma points via the measurement model and calculate the anticipated measurement
  - Calculate the Kalman gain from the measurement covariance and cross-covariance
  - Compute the updated mean and covariance
- It ends up still being the same five lines as the EKF where you take $\mathcal{X}_t(\dots)$ to be the function that calculates the sigma points

Prediction:

$$\bar{\mathcal{X}}_t = f(u_t, \mathcal{X}(\mu_{t-1}, \Sigma_{t-1}), 0)$$

$$\bar{\mu}_t = \sum_{i=0}^{2d} w_m^{[i]} \bar{\mathcal{X}}_t$$

$$\bar{\Sigma}_t = \sum_{i=0}^{2d} w_c^{[i]} \left(\bar{\mathcal{X}}_t^{[i]} - \bar{\mu}_t\right)\left(\bar{\mathcal{X}}_t^{[i]} - \bar{\mu}_t\right)^T + Q_t$$

Update:

$$\bar{\mathcal{Z}}_t = g\left(\mathcal{X}(\bar{\mu}_t, \bar{\Sigma}_t)\right)$$

$$\hat{z}_t = \sum_{i=0}^{2d} w_m^{[i]} \bar{\mathcal{Z}}_t^{[i]}$$

$$S_t = \sum_{i=0}^{2d} w_c^{[i]} \left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right)\left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right)^T + R_t$$

$$\hat{\Sigma}_t = \sum_{i=0}^{2d} w_c^{[i]} \left(\mathcal{X}_t^{[i]} - \bar{\mu}_t\right)\left(\mathcal{Z}_t^{[i]} - \hat{z}_t\right)^T$$

$$K_t = \hat{\Sigma}_t S_t^{-1}$$

$$\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$$

$$\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^T$$

Assumptions: normal distribution assumption is relaxed, but still relies on it being a reasonably accurate approximation

- Strengths
  - Simple, has the strengths of the Kalman Filter with added flexibility
  - Works with generic models
  - Very computationally efficient, sums can be vectorized, and works well even for higher dimensional systems
- Limitations and weaknesses
  - Limited to systems with lower-order nonlinearities, but is more flexible than the EKF
  - Still limited to unimodal distributions

The EKF versus the UKF

The EKF is more frequently used for nonlinear systems.

- The Kalman Filter is old (original research published from 1959-1961) and the EKF (1970's and 1980's) is a straightforward step forward from there using old and well-established linearization techniques (Taylor Series: 1715).

- More frequently encountered in industry and the literature simply due to the fact that it is well understood and people have practice with it
- Nonetheless it has notable shortcomings, is difficult to implement and tune, and reliable only in certain well constrained systems (Julier, S.J.; Uhlmann, J.K. (2004). "Unscented filtering and nonlinear estimation")

The UKF is relatively new (1997-2004) and not widely adopted

- In my opinion, while the notation is a bit intimidating, the UKF ends up being easier to implement
- Has been found to be more accurate than the EKF, but EKFs that use higher-order series-expansions have been shown to recover that accuracy.

Nonparametric Filters

What if we don't assume a normal distribution?

Bayes' Filter

- Prior: $p(x_0)$
- Transition model: $f(x_t|x_{t-1},\ u_t)$
- Measurement model: $g(z_t|x_t)$
- Prediction step:

$$p(x_t|z_{1:t-1},\ u_{1:t}) = \int f(x_t|x_{t-1},\ u_t)\ p(x_{t-1}|z_{1:t-1},\ u_{1:t-1})dx_{t-1}$$

- Update step:
- $p(x_t|z_{1:t}, u_{1:t}) = \frac{g(z_t|x_t)p(x_t|z_{1:t-1},\ u_{1:t})}{\int g(z_t|\ \bar{x}_t)\ p(\ \bar{x}_t|z_{1:t-1}, u_{1:t})d\bar{x}_t}$

Instead of a normal distribution, instead represent the prior state using some finite number of values.

Retain the arbitrary/generic process model $f(x_t|x_{t-1},\ u_t)$ and measurement model $g(z_t|x_t)$.

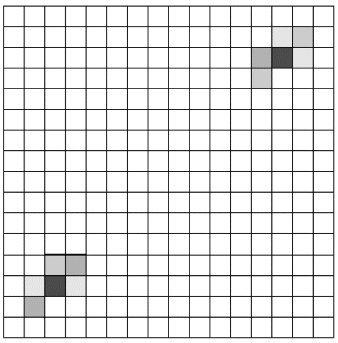
Use this to address the primary disadvantages of Kalman Filter based techniques

- No longer limited to linear and Gaussian
- Can now handle multi-modal distributions or multiple hypothesis

State is now represented by some discrete sub-sample of the continuous state space (grid cell, voxel, particle, etc.).

Histogram Filter

A histogram filter decomposes the state space into a finite number of regions which bound a section of the continuous state space



- Ex: a 1x1 meter 2D Cartesian grid, seen on the right
- x_k denotes region (or cell) k

Region properties:

- Convex
- Disjoint ($x_i \cap x_k = 0$)
- Span the state space
- Each has a single probability value: $p_i = p(x \in x_i)$

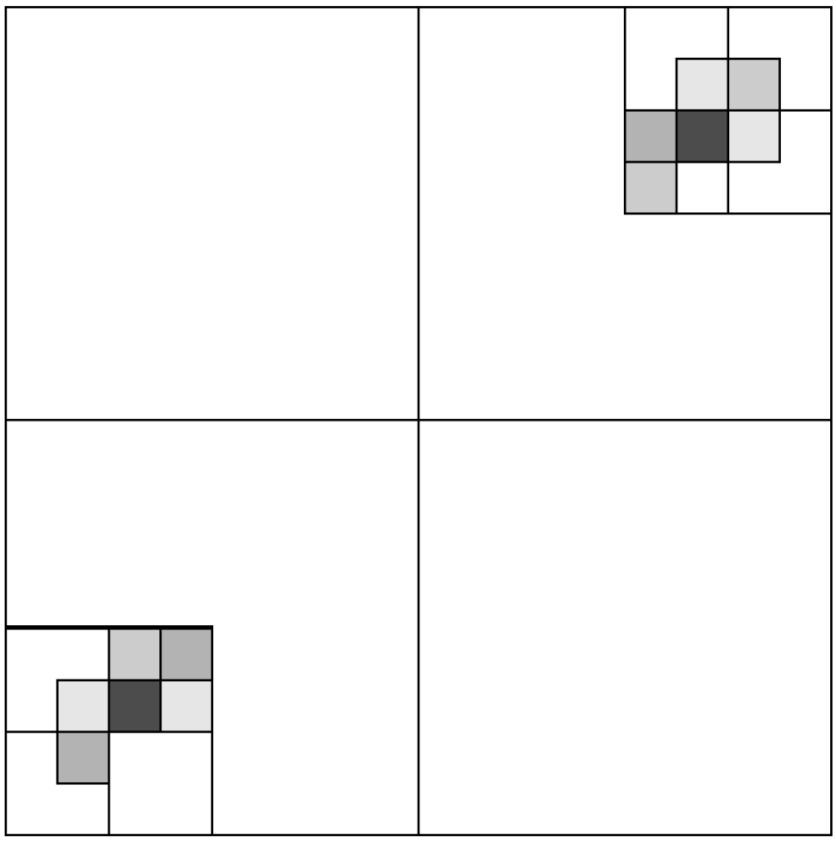
Uniform grid

Simple and straightforward implementation

- All cells are the same size
- Cell dimensions don't change

Advantages

- Simple, easy to implement



Disadvantage

- Tradeoff between accuracy and computation speed
- More accurate (detailed) grids require more computation

Quad tree

Grid-tree based technique that progressively sub-divides cells into four equally sized cells

- Starts with a small number of very large cells
- High probability cells are divided

Advantages

- More computationally efficient
- Balances accuracy with speed

Disadvantages

- Algorithmically complex
- Relating a tree structure to grid structure is not especially intuitive

Process Model

Problem: process models ($p(x_t|x_{t-1},\ u_t)$) are defined in terms of specific state values

- Probability of transitioning from *state* to *state* given an input
- Histogram filter represents the state as a *region*
- Need to find a region-based process model that is analogous

If the grid cells are small enough, we can approximate the region by its center point

- $\hat{x}_{i,t} = \frac{\int_{x_{i,t}} x\, dx}{\int_{x_{j,t}} dx}$
- $p(x_{k,t}|x_{i,t-1},\ u_t) \approx \gamma|x_{i,t}|p(\hat{x}_{k,t}|\hat{x}_{i,t-1}, u_t)$
- In English: the probability value of the grid cells at time t given the prior probability values and the control input is approximately equal the prior values multiplied the probability of transitioning from the old cell to the new cell
- γ is a normalizer to ensure the probability distribution sums to one.

Observation Model

Same problem as the process model: models are typically defined in terms of specific state values not regions.

Need to calculate a region-based probability of receiving measurement z_t from the region $x_{k,t}$.

Use the same center-point approximation: $\hat{x}_{k,t} \approx x_{k,t}$

- $p(z_t|x_{k,t}) \approx p(z_t|\hat{x}_{k,t})$

There is no universal rule or limit for the validity of the center point assumption. It's a trade off:

- Accuracy of navigation solution
- Computational resources
- Accuracy of sensors
- Performance demands

Pseudocode

Prediction

for $i = 1{:}M$

for $k = 1{:}\,M$

$p_{k,i} = |x_{k,t}|p(\hat{x}_{k,t}|\hat{x}_{i,t-1}, u_t)$

$\gamma_i = \sum_{k=1}^{M} p_{k,i}$

for $k = 1{:}\,M$

$\bar{p}(x_{k,t}|u_{1:t},\ z_{1:t-1}) = \sum_{k=1}^{M} \frac{p_{k,i}}{\gamma_i}$

Update

for $k = 1{:}\,M$

$p_k = g(z_t|\hat{x}_{k,t})\bar{p}(x_{k,t}|u_{1:t}, z_{1:t-1})$

$\gamma = \sum_{k=1}^{M} p_k$

for $k = 1{:}\,M$

$p(x_{k,t}|u_{1:t},\ z_{1:t}) = \frac{p_k}{\gamma}$

One-dimensional example and comparison

Stated-based

Process model

- $x_t = x_{t-1} + u_t + \eta_t$
- $Q_t = 0.5$

Observation model

- $z_t = x_t + \nu_t$
- $R_t = 1.0$

Region Based

State representation:

- 10 meters
- Half-meter bins ($|x_{k,t}| = 0.5$)
- Bin edges: $x_k = [(0.5k \quad 0.5(k+1))$ for $k = 0{:}\,20]$

Center points are $\hat{x}_k = 0.25 + 0.5k$

Approximate process model

- $p(\hat{x}_{k,t}|\hat{x}_{i,t-1}, u_t) = \mathcal{N}(\hat{x}_{k,t} - (\hat{x}_{i,t-1} + u_t),\ 0,\ 0.5)$
- $p(x_{k,t}|x_{i,t-1}, u_t) \approx \gamma\ 0.5\ \mathcal{N}(0.5(k-i) - u_t,\ 0,\ \ 0.5)$

Approximate observation model

- $p(z_t|x_{k,t}) \approx p(z_t|\hat{x}_{k,t}) \approx \mathcal{N}(z_t - (0.25 + 0.5k),\ 0,\ 1)$

The Particle Filter

Like the Histogram Filter: instead of a normal distribution, instead represent the prior state using some finite number of values.

Retain the arbitrary/generic process model $f(x_t|x_{t-1},\ u_t)$ and measurement model $g(z_t|x_t)$.

State is now represented by a finite number of discrete hypotheses (particles): $x^{[i]}$

- Number of particles (typically N, M, or k) typically needs to be large
- Small-scale navigation problems can get away with a few hundred
- Larger applications (self-driving cars, outdoor drones) typically require thousands
- Marine/aerospace navigation (trans-oceanic navigation, ballistic missiles, orbital satellites) typically require more that 10,000

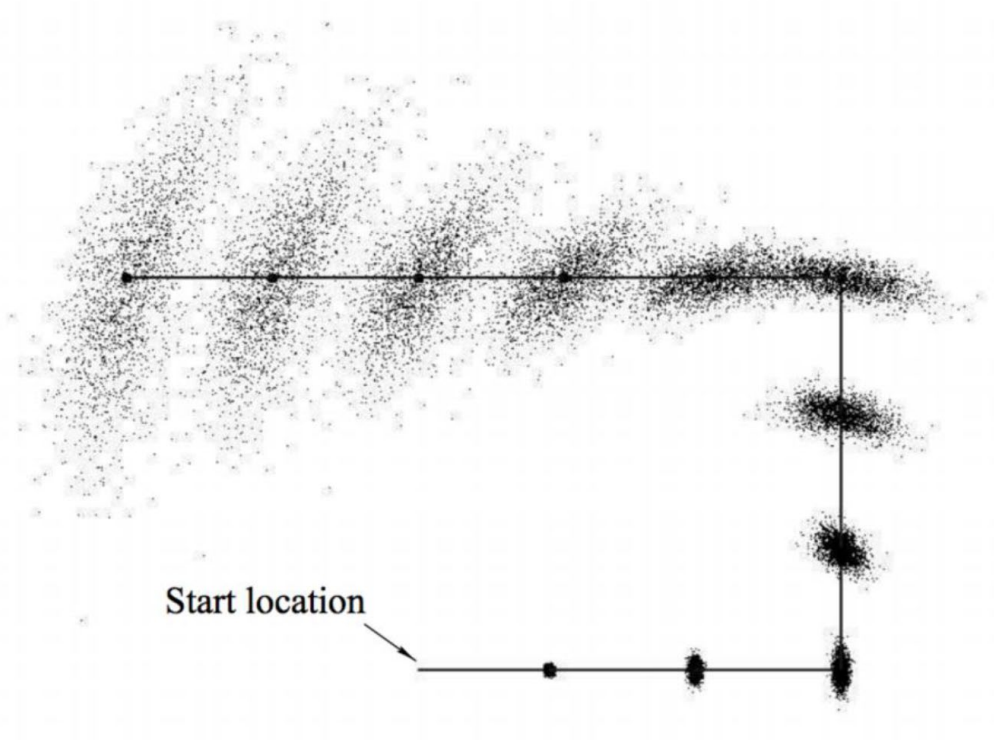
Random "Monte Carlo" approach: for each particle in the prior draw a new particle according to the process model

$$x_t^{[i]} \sim \left(x_t \middle| x_{t-1}^{[i]},\ u_t\right)$$

This 'randomly' moves each particle

In practice, we deterministically propagate each particle according to the known physical model and add in stochastic noise.

- Noise element typically from the noise values on the control inputs
- Typically, velocities/accelerations so the noisiness of the IMU plays a part here



The particle filter refines the uncertainty of the distribution by re-weighting and resampling the particle distribution

Give each particle a new weight according to the measurement model

- $w^{[i]} = g\left(z_t \middle| x_t^{[i]}\right)$
- Particles that more accurately estimate the true state will receive higher weights

After reweighting, resample the particles in proportion to their weight

- Many resampling methods
- May or may not increase or decrease the particle count
- For our purposes, we'll limit the particle filter to a set particle filter count

Need enough particles, very much a manual tuning process.

- More particles *tends to* increase accuracy
- More particles increases computational load

Resampling introduces randomness, exact repeatability is hard to achieve.

Particle deprivation can result in the particle distribution not covering truth

- Can lead to navigation solution divergence (inaccuracy)
- Can lead to a local minima navigation solution where the filter gets stuck on a close but inaccurate solution

Resampling assumes weights coming from the observation model closely match the true underlying distribution

- Can lead to bad results if no sample covers truth
- Can mitigate by adding noise to the observation model

Resampling

Resampling is critical step to reduce uncertainty in the particle filter

The new set of particles is drawn from the prior set with probability proportional to the importance weighting

In practice, the importance weighting is frequently normalized to more accurately reflect a probability

The same particle can be resampled multiple times.

Many different methods that draw from random sampling theories.

Multinomial

Randomly draw each new particle with probability proportional to its weight

$$w_{total} = \sum_{i=1}^{M} w^{[i]}$$

$$X = [\square]$$

$$\text{for } i = 1{:}M$$

$r = \text{rand}(0,\ w_{total})$

$j = \text{find}(w > r, \text{"first"})$

$\text{append}\left(x^{[j]}\right) \rightarrow X$

Low Variance Resampling

Draw a single random number and use that to select all the particles.

Has the advantage of more efficient code (single random number) and ends up sampling the state space better.

$$w_{total} = \sum_{i=1}^{M} w^{[i]}$$

$$r = \text{rand}\left(0, \frac{w_{total}}{M}\right)$$

$$X = [\square];\ i = 1;\ c = w^{[i]};$$

$$\text{for } i = 1{:}\,M$$

$u = r + \frac{(k-1)w_{total}}{M}$

$\text{while } u > c$

$i++$

$c \mathrel{+}= w^{[i]}$

$\text{append}\left(x^{[i]}\right) \rightarrow X$

Other Methods

Residual Resampling

- The number of particles selected for each particle is proportional to its importance weight, but without replacement. After resampling, some particles may be duplicated.
- This method is more efficient than multinomial resampling when there are particles with very high importance weights.

Stratified Resampling

- Divides the unit interval into strata and ensures that each stratum contains at least one particle.
- Particles are selected randomly from each stratum, reducing the chance of sampling a high-weight particle multiple times.

Systematic Resampling:

- More deterministic method, sorts the particles based on their cumulative importance weights and then selects particles at regular intervals.
- Good balance between maintaining diversity and computational efficiency.

Many others, generally seek to address a specific computation or particle degeneracy problem.

Recovery from failure

Particle deprivation can be a serious problem when using particle filters.

Occasionally inject new particles into the system to avoid particle deprivation

- Need to do this more when there is more uncertainty
- Can potentially be done at every resample

Can add particles in different ways

- Uniformly at random
- That agree with the measurements

Example

Let's go back to our two-dimensional, three degree of freedom example.

- For some simplicity, let's assume we can control our linear and angular velocity
- $x_t = \begin{bmatrix} p_x \\ p_y \\ \theta \end{bmatrix}$ and $u_t = \begin{bmatrix} v \\ \omega \end{bmatrix}$
- We again have noise on the IMU that measures our commanded velocities $\eta = \begin{bmatrix} \eta_v \\ \eta_\omega \end{bmatrix}$

Now let's formulate our process model to *include* noise.

State transition equation is now:

$$\dot{x}_t = f(x_t,\, u_t, \eta_t) = \begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\hat{v}}{\hat{\omega}}(\sin(\theta - \hat{\omega}t) - \sin\theta) \\ \frac{\hat{v}}{\hat{\omega}}(\cos\theta - \cos(\theta + \hat{\omega}t)) \\ \hat{\omega}t \end{bmatrix}$$

Where $\hat{v}$ and $\hat{\omega}$ are drawn from the normal distributions $\mathcal{N}(v,\, \eta_v)$ and $\mathcal{N}(\omega,\, \eta_\omega)$ respectively

- v and ω are the commanded (control) quantities.
- $\hat{v}$ and $\hat{\omega}$ stochastically account for any difference between the commanded control values and those measured by the IMU

Particles are then re-weighted according to the measurement model $g\left(z_t \middle| x_t^{[i]}\right)$.

Particles are then resampled according to their weights.

Initialize X_{t-1}, where $x_{t-1}^{[i]} \in X_{t-1}$ and $x_{t-1}^{[i]} \in \mathcal{S}^d$, $i \leq M$

$$\bar{X}_t = [\,]$$

For $m = 1{:}\, M$

$x_t^{[m]} = f\left(x_{t-1}^{[m]},\, u_t, \eta_t\right)$

$w_t^{[m]} = g\left(z_t \middle| x_t^{[m]}\right)$

For $m = 1{:}\, M$

sample $i \sim w_t$

append $x_t^{[i]} \rightarrow \bar{X}_t$

Return $\bar{X}_t$