Nonparametric Filters

ALTERNATIVES TO GAUSSIAN MODELING

Introduction

WHAT IF WE DON'T ASSUME A NORMAL DISTRIBUTION?

Start with Bayes

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:

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

Assumptions

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 REGION-BASED ESTIMATOR

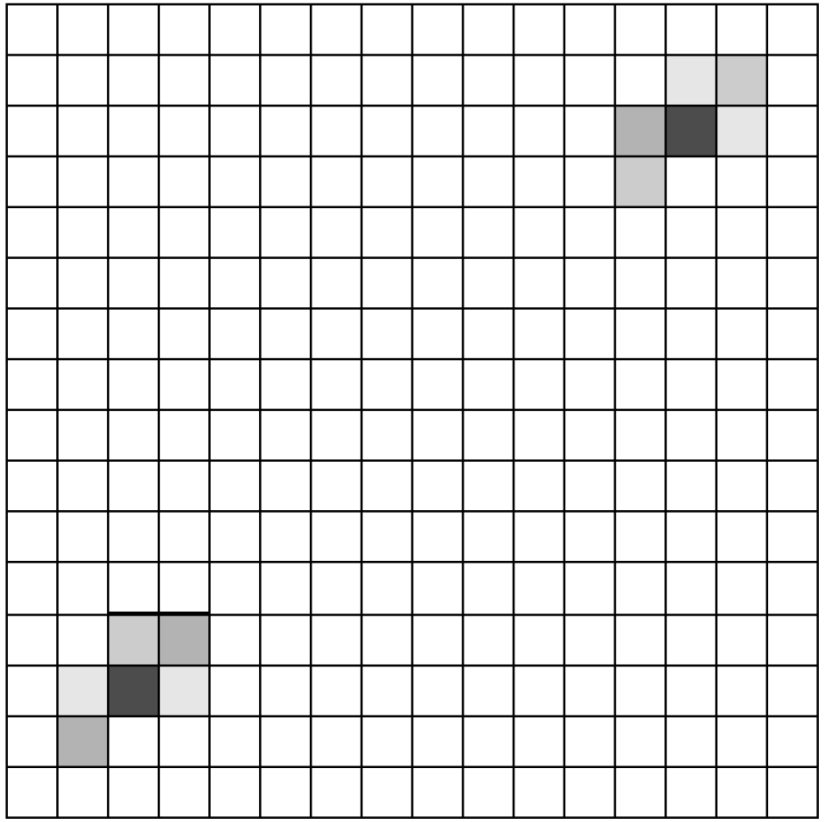
State Representation

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: $p_i = p(x \in x_i)$



Decomposition: Uniform Grid

Simple and straightforward implementation

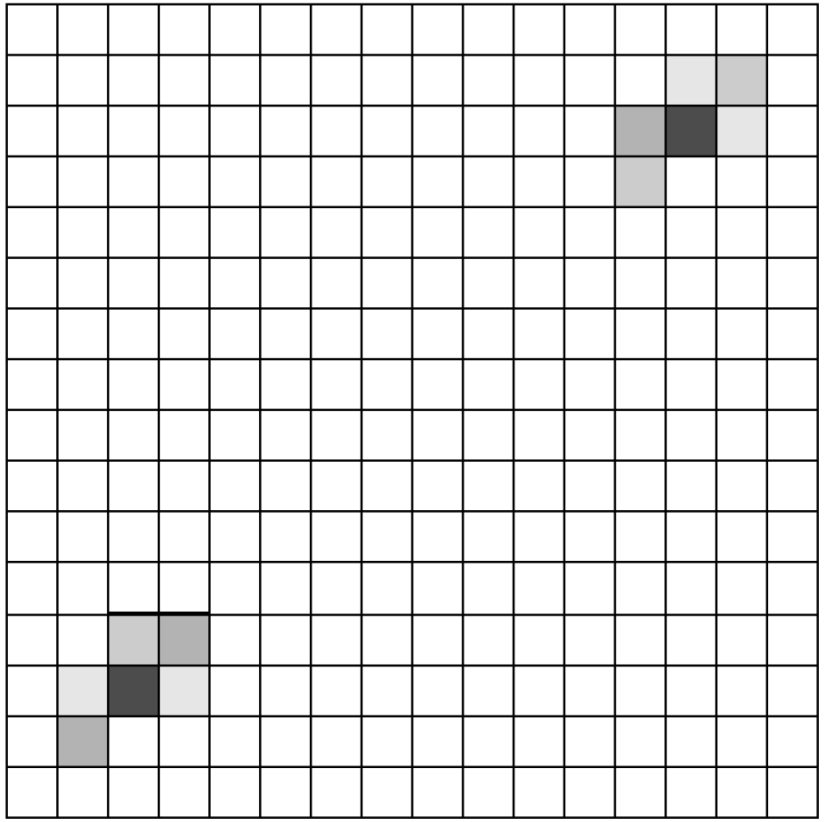
- All cells are the same size
- Cell dimensions don’t change

Advantages

- Simple, easy to implement

Disadvantage

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



Decomposition: Quadtree

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

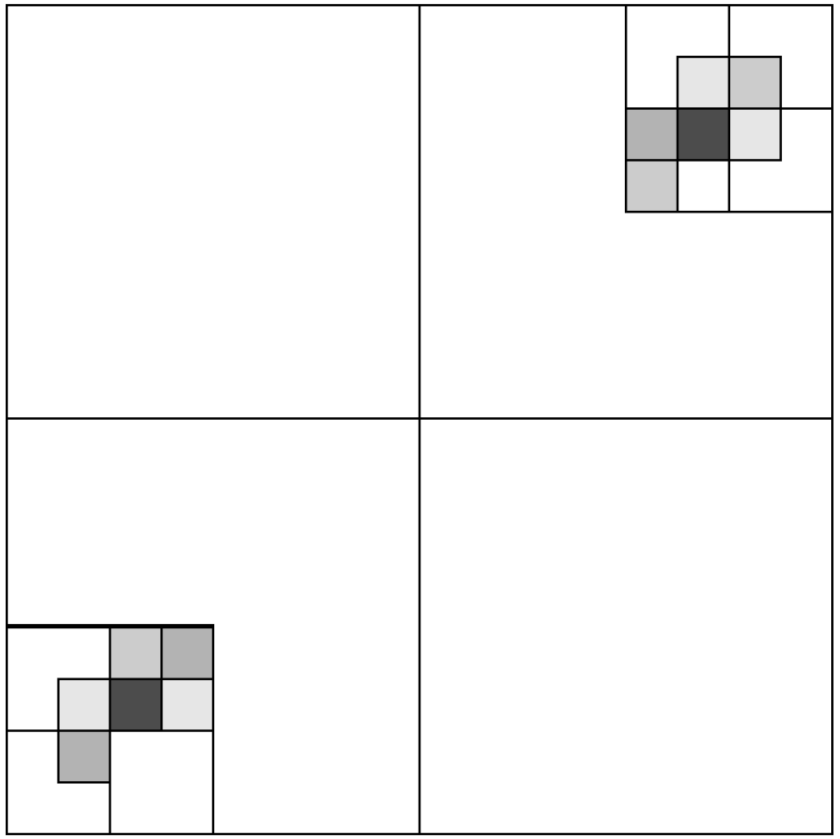
- Starts with a small number of very large cells
- High probability cells are divided

Advantages

- More computationally efficient
- Balances accuracy with speed

Disadvantages

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



Histogram Filter: 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.

Histogram Filter: 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

Histogram Filter: Pseudocode

PREDICTION

for $i = 1: M$

  for $k = 1: M$

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

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

for $k = 1: M$

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

UPDATE

for $k = 1: M$

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

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

for $k = 1: M$

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

Histogram Filter: A 1D Example

STATE-BASED MODEL

Process model

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

REGION-BASED MODEL

State representation:

- Half-meter bins ($|x_{k,t}| = 0.5$)
- $x_k = [0.5(k-1), 0.5k)$ for $k = 1{:}\,16$

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

Approximate process model

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

Histogram Filter: A 1D Example

STATE-BASED MODEL

Observation model

- $z_t = x_t + v_t$
- $R_t = 1.0$

REGION-BASED MODEL

Approximate observation model

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

Particle Filter

A DISCRETE STATE-BASED ESTIMATOR

Assumptions

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

Particle Filter: Prediction Step

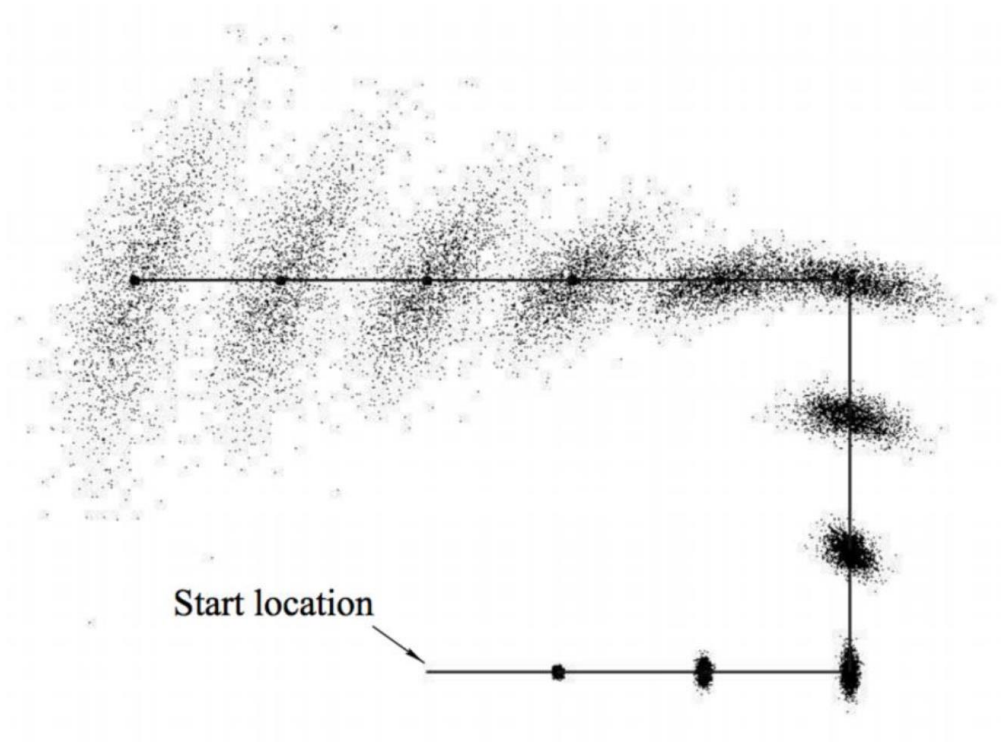
Random “Monte Carlo” approach: for each particle in the prior draw a new particle according to the process model

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

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



Particle Filter: Update Step

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

Particle Filter: Limitations

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

Particle Filter: 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.

Basic multinomial (naïve) resampling

Randomly draw each new particle with probability proportional to its weight

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

$$X = [\quad]$$

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

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

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

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

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.

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

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

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

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

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

$$\quad \text{while } u > c$$

$$\quad\quad i++$$

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

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

Other resampling 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

Particle Filter: An 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.

Particle Filter: An Example

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.

Particle Filter: Pseudocode

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

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

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 $\bar{X}_t$