2. Bayes and Kalman Filters

Bayes Filter

- The Bayes Filter forms the **foundation** for all other filters in this class
  - As described in background slides, Bayes rule is the right way to incorporate new probabilistic information into an existing, **prior estimate**
  - The resulting filter definition can be implemented **directly for discrete state systems**
  - For continuous states, need additional assumptions, additional structure to **solve the update equations analytically**

Formulation

- State: x_i
  - All aspects of the vehicle and its environment that can impact the future
  - **Assume the state is complete**
- Control inputs: u_t
  - All elements of the vehicle and its environment that can be controller
- Measurements: y_t
  - All elements of the vehicle and its environment that can be sensed
- Notaiton:
  - Discrete time index t
  - Initial state is x_0
  - First, apply control action u_1
  - Move to state x_1
  - Then, take measurement y_1

Motion Modeling

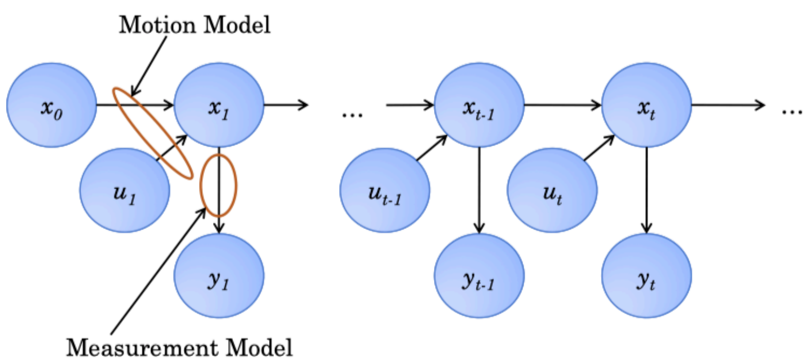
- **Complete state:**
  - At each time t, x_{t-1} is a sufficient summary of all previous inputs and measurements:
    - $p(x_t | x_{0:t-1}, y_{1:t-1}, u_{1:t}) = p(x_t | x_{t-1}, u_t)$
  - Application of Conditional Independence
    - No additional information is to be had by considering previous inputs or measurements
  - Referred to as the **Markov Assumption**
    - Motion model is a **Markov Chain**

Measurement Modeling

- Complete state:
  - Current state is sufficient to model all previous states, measurements and inputs:
    - $p(y_t | x_{0:t}, y_{1:t-1}, u_{1:t}) = p(y_t | x_t)$
- Again, conditional independence
- Recall, in standard LTI state space model, measurement model **may also depend on the current input**

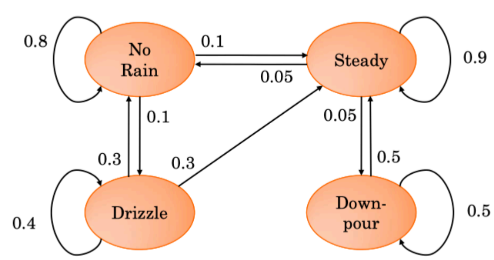
Combined Model

- Referred to as **Hidden Markov Model (HMM)** or **Dynamic Bayes Network (DBN)**



[Example] Discrete State Motion & Measurement Model:

- Example Motion Model:
  - States: $\{NoRain, Drizzle, Steady, Downpour\}$
  - Inputs: **None**



  - For discrete states, the motion model can be written in matrix form
    - For each input u_t , the $n \times n$ motion model matrix is
      - $p(x_t | u_t = u, x_{t-1}) = \begin{bmatrix} p(x_t = x_1 | x_{t-1} = x_1) & p(x_t = x_1 | x_{t-1} = x_2) & \dots \\ p(x_t = x_2 | x_{t-1} = x_1) & p(x_t = x_2 | x_{t-1} = x_2) & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$
      - $\downarrow$ Each **row** — defines the probabilities of transitioning to state x_t from all possible states x_{t-1}
      - $\rightarrow$ Each **column** $|$ defines the probabilities of transitioning to any state x_t from a specific state x_{t-1}
      - Again, the columns must sum to 1 : $\sum_{|} p_i = 1$
    - Example:

- $$p(x_t|u_t = u, x_{t-1}) = \overbrace{\begin{bmatrix} 0.8 & 0.3 & 0.05 & 0 \\ 0.1 & 0.4 & 0 & 0 \\ 0.1 & 0.3 & 0.9 & 0.5 \\ 0 & 0 & 0.05 & 0.5 \end{bmatrix}}^{x_{t-1}} \Big\} x_t$$

- Example Measurement Model:
  - States: $\{NoRain, Drizzle, Steady, Downpour\}$
  - Measurements: $\{Dry, Light, Medium, Heavy\}$
    - $$p(x_t|u_t = u, x_{t-1}) = \overbrace{\begin{bmatrix} 0.95 & 0.1 & 0 & 0 \\ 0.05 & 0.8 & 0.15 & 0 \\ 0 & 0.1 & 0.7 & 0.1 \\ 0 & 0 & 0.15 & 0.9 \end{bmatrix}}^{x_t} \Big\} y_t$$
    - Again, the columns must sum to 1 : $\sum_{|} p_i = 1$

Aim of Bayes Filter

1. To estimate the current state of the system based on all known inputs and measurements.
   - That is, to define **a belief** about the **current state** using all **available** information:
     - $\overline{bel}(x_t) = p(x_t|y_{1:t}, u_{1:t})$
   - Known as **belief**, state of knowledge, information state Depends on every bit of information that exists up to time t
2. Can also **define a belief prior** to **measurement** y_t
   - $\overline{bel}(x_t) = p(x_t|y_{1:t-1}, u_{1:t})$
   - Known as **prediction**, **predicted state**

Bayes Filter | Problem Formulation

- Given a **prior** for the system state:
  - $p(x_0)$
- Given **motion** and **measurement models**:
  - $\overbrace{p(x_t|x_{t-1}, u_t)}^{motion} \qquad \overbrace{p(yt|xt)}^{measurement}$
- Given a sequence of inputs and measurements:
  - $u_{1:t} = \{u_1, \ldots, u_t\}, \qquad y_{1:t} = \{y_1, \ldots, y_t\}$
- Estimate the current state distribution
  - (Form **a belief** about the current state):
  - $bel(x_t) = p(x_t|y_{1:t}, u_{1:t})$

Bayes Filter | Algorithm

[Bayes Filter | Algorithm Abstract]

- At each time step, t, for all possible values of the state x:

1. Prediction Update : (*Total Probability*)

- $\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$

2. Measurement Update : (*Bayes Theorem*)

- $bel(x_t) = \eta\, p(y_t|x_t)\overline{bel}(x_t)$
- η : normalizing constant
  - does not depend on the state
- Recursive estimation technique

[Recall | Bayes Filter Theorem]

- $p(a|b) = \frac{p(b|a)\, p(a)}{p(b)}$
- **Terminology:**
    - $posterisor = \frac{likelihood \cdot prior}{evidence}$

[Derivation | Proof by Induction]

- Demonstrate that belief at time t can be found using
    - belief at time $t-1$,
    - input at t
    - and measurement at t
- Initially:
    - $bel(x_0) = p(x_0)$
- At time t, from [Recall] Bayes Filter Theorem relates x_t, y_t

$$
\begin{aligned}
bel(x_t) &= p(x_t|y_{1:t}, u_{1:t}) = p(x_t|y_t, y_{1:t-1}, u_{1:t}) \\
&= \frac{p(y_t|x_t, y_{1:t-1}, u_{1:t}) p(x_t|y_{1:t-1}, u_{1:t})}{p(y_t|y_{1:t-1}, u_{1:t})} \\
&= \overbrace{p(y_t|x_t, y_{1:t-1}, u_{1:t})}^{p(y_t|x_t)\text{: measurement model.}} \quad \overbrace{p(x_t|y_{1:t-1}, u_{1:t})}^{\overline{bel}(x_t)\text{: belief prediction.}} \quad \overbrace{p(y_t|y_{1:t-1}, u_{1:t})^{-1}}^{\eta\text{: const. normalizer}} \\
&= \eta\, p(y_t|x_t)\, \overline{bel}(x_t)
\end{aligned}
$$

- Hence:
    - **Measurement Update** $\equiv bel(x_t) = \eta\, p(y_t|x_t)\, \overline{bel}(x_t)$
        - Multiplication of two vectors
        - Requires the measurement y_t to be known
- However, we now need to find the **belief prediction**:
    - Done using total probability over previouse state:

    $$
    \begin{aligned}
    \overline{bel}(x_t) &= p(x_t|y_{1:t-1}, u_{1:t}) \\
    &= \int \overbrace{p(x_t|x_{t-1}, y_{1:t-1}, u_{1:t})}^{\text{a) can incorporate motion model}} \; \overbrace{p(x_{t-1}|y_{1:t-1}, u_{1:t})}^{\text{b) = bel(x\_\{t-1\})}} \, dx_{t-1}
    \end{aligned}
    $$

    a) Incorporating Motion Modeling:

    $$p(x_t|x_{t-1}, y_{1:t-1}, u_{1:t}) = p(x_t|x_{t-1}, u_t)$$

    b) And we note that the control input at time t does not affect the state at time $t-1$:

    $$p(x_{t-1}|y_{1:t-1}, u_{1:t}) = p(x_{t-1}|y_{1:t-1}, u_{1:t-1}) = bel(x_{t-1})$$

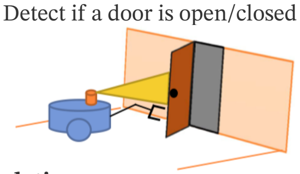
    - Hence:
        - **Prediction Update** $\equiv \overline{bel}(x_t) = \int p(x_t|x_{t-1}, u_t)\, bel(x_{t-1})\, dx_{t-1}$
            - Summative over discrete states
- END OF **proof by induction** for Bayes Filter Algorithm
    - For this step, we need the control input to define the correct motion model distribution

Summary:

- If **state, measurements, inputs** are **DISCRETE**,
    - can **directly implement *Bayes Filter***
        - **Prediction update** is ***summation*** over **discrete** states
        - **Measurement update** is ***multiplication*** of **two vectors**
- Else, if they are **CONTINUOUS**
    - must define model or approximation to enable computation:
        - **KF** (Kalman Filter)
            - **Linear** motion models
            - **Linear** measurement models
            - **Additive Gaussian** disturbance and noise distributions

- **EKF / UKF** (Extended Kalman Filter / Unscented Kalman Filter)
  - **Non-linear** motion models
  - **Non-linear** measurement models
  - **Additive Gaussian** disturbance and noise distributions
- **PF** (Particle Filter)
  - **(Dis)continuous** motion models
  - **(Dis)continuous** measurement models
  - **General** disturbance and noise distributions

[Example 1 | Discrete Bayes Filter]

- **Problem:**
  - Detect if a door is open/closed with a robot that can sense the door position and pull the door open
  - 
- **Formulation:**
  - **State**: $door = \{open, closed\}$
  - **State Prior** (uniform):
    - $p(x_0) = \begin{cases} p(open) = 0.5 \\ p(closed) = 0.5 \end{cases}$
  - **Inputs**: $arm\_command = \{none, pull\}$
  - **Motion Model**:
    - If input = $none$, do nothing:
      - $p(x_t|u_t = none, x_{t-1}) \rightarrow \begin{cases} p(open_t|none, open_{t-1}) = 1 \\ p(closed_t|none, open_{t-1}) = 0 \\ p(open_t|none, closed_{t-1}) = 0 \\ p(closed_t|none, closed_{t-1}) = 1 \end{cases}$
    - If input = $pull$, pull the door open:
      - $p(x_t|u_t = pull, x_{t-1}) \rightarrow \begin{cases} p(open_t|none, open_{t-1}) = 1 \\ p(closed_t|none, open_{t-1}) = 0 \\ p(open_t|none, closed_{t-1}) = 0.8 \\ p(closed_t|none, closed_{t-1}) = 0.2 \end{cases}$
  - **Measurements**: $meas = \{sense\_open, sense\_closed\}$
  - **Measurement model** (noisy door sensor):
    - $p(y|x) \rightarrow \begin{cases} p(sense\_open|open) = 0.6 \\ p(sense\_open|closed) = 0.2 \\ p(sense\_closed|open) = 0.4 \\ p(sense\_closed|closed) = 0.4 \end{cases}$
- **Example:**
  - **At** $t = 1$,
    - input $u_1 = none$:
      - 1. **Prediction Update** : (***Total Probability***):
        - $\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$
        - $\overline{bel}(x_1) = \int p(x_1|u_1, x_0) bel(x_0) dx_0 = \sum p(x_1|u_1, x_0) p(x_0)$
      - Calculate belief prediction for each possible value of state:
        - $\overline{bel}(open_1) = p(open_1|none_1, open_0) bel(open_0) + p(open_1|none_1, closed_0) bel(closed_0) \\ = 1 * 0.5 + 0 * 0.5 = 0.5$
        - $\overline{bel}(closed_1) = p(closed_1|none_1, open_0) bel(open_0) + p(closed_1|none_1, closed_0) bel(closed_0) \\ = 0 * 0.5 + 1 * 0.5 = 0.5$
    - measurement $y_1 = sense\_open$:
      - 2. **Measurement Update** : (***Bayes Theorem***):

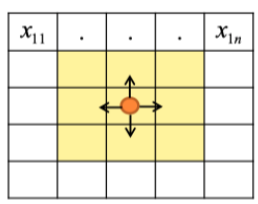
- $bel(x_1) = \eta\, p(y_1|x_1)\overline{bel}(x_1)$
- Calculate for each possible value of state:
  - $$\begin{aligned} bel(open_1) &= \eta\, p(sense\_open_1|open_1)\overline{bel}(open_1) \\ &= \eta 0.6 \cdot 0.5 = 0.3\eta \end{aligned}$$
  - $$\begin{aligned} bel(closed_1) &= \eta\, p(sense\_open_1|closed_1)\overline{bel}(closed_1) \\ &= \eta 0.2 \cdot 0.5 = 0.1\eta \end{aligned}$$
- Caluclate **normalizer** η and solve for **posterior**
  - $\eta = \frac{1}{0.3+0.1} = 2.5$
  - $\Rightarrow \begin{aligned} bel(open_1) = 0.75 \\ bel(closed_1) = 0.25 \end{aligned}$

- **At** $t = 2$,
  - With $u_2 = pull, \quad y_2 = sense\_open$
  - Then:
    - State propagation:
      - $\overline{bel}(open_2) = 1 \times 0.75 + 0.8 \times 0.25 = 0.95$
      - $\overline{bel}(closed_2) = 0 \times 0.75 + 0.2 \times 0.25 = 0.05$
    - Measurement Update:
      - $bel(open_2) = \eta 0.6 \cdot 0.95 = 0.983$
      - $bel(closed_2) = \eta 0.2 \cdot 0.05 = 0.017$

- **In summary:**
  - Uniform prior, do nothing, measure open: $bel(open_1) = 0.75$
  - Pull open, measure open: $bel(open_2) = 0.983$

[Example 2 | Histogram Filter]

- Motion of robot in a $n \times n$ grid:



  - **State:**
    - Position $= \{x_{11}, x_{12}, \ldots, x_{1n}, \ldots, x_{nn}\}$
  - **Input**:
    - Move = $\{Up, Right, Down, Left\}$
    - 40% chance the move does not happen
    - Cannot pass through outer walls
  - **Measurement**: Accurate to within 3×3 grid
    - $p(y(i-1:i+1, j-1:j+1)|x(i,j) = \begin{bmatrix} .11 & .11 & .11 \\ .11 & .12 & .11 \\ .11 & .11 & .11 \end{bmatrix}$
  - **Prior over states**
    - Assume no information, uniform
    - Vector of length n^2
      - $p(x_0) = \frac{1}{n^2}$
  - **Motion Model:**
    - Given a particular input and previous state, probability of moving to any other state
      - $n \times n$ state, one for each grid point
      - 4 input choices

        - $p(x_t|x_{t-1}, u_t) \in [0, 1]^{n^2 \times n^2 \times 4}$
    - Code:

```
mot_mod = zeros(N,N,4);

for i=1:n
  for j=1:n
    cur = i+(j-1)*n;

    % Move up
    if (j > 1)
      mot_mod(cur-n,cur,1) = 0.6;
      mot_mod(cur,cur,1) = 0.4;
    else
      mot_mod(cur,cur,1) = 1;
    end

    % Move right
    if (i < n)
      mot_mod(cur+1,cur,2) = 0.6;
      mot_mod(cur,cur,2) = 0.4;
    else
      mot_mod(cur,cur,2) = 1;
    end
  end
end
```

  - **Measurement Model**
    - Given any current state, probability of a measurement
    - Same number of measurements as states $p(y_t|x_t) \in [0, 1]^{n^2 \times n^2}$
    - Same 3×3 matrix governs all interior points
    - Boundaries cut off invalid measurements and require normalization
    - Very simplistic and bloated model
        - Could replace with 2 separate states and measurements to perpendicular walls
    - Code:

```
%% Create the measurement model

meas_mod_rel = [0.11 0.11 0.11;
                0.11 0.12 0.11;
                0.11 0.11 0.11];

% Convert to full measurement model
% p(y_t | x_t)
meas_mod = zeros(N,N);

% Fill in non-boundary measurements
for i=2:n-1
  for j=2:n-1
    cur = i+(j-1)*n;
    meas_mod(cur-n+[-1:1:1],cur) = meas_mod_rel(1,:);
    meas_mod(cur+[-1:1:1],cur) = meas_mod_rel(2,:);
    meas_mod(cur+n+[-1:1:1],cur) = meas_mod_rel(3,:);
  end
end
```

- Making moves:

```
videoobj=VideoWriter('bayesgrid.mp4','MPEG-4');
truefps = 1; videoobj.FrameRate = 10; %Anything less than 10 fps fails. open(videoobj);

figure(1);clf; hold on;
beliefs = reshape(bel,n,n);
imagesc(beliefs);
plot(pos(2),pos(1),'ro','MarkerSize',6,'LineWidth',2)
colormap(bone);
title('True state and beliefs')
F = getframe;

% Dumb hack to get desired framerate
for dumb=1:floor(10/truefps)
  writeVideo(videoobj, F);
end
```

- Simulation Code:

```
%Main Loop
for t=1:T
  %% Simulation
  % Select motion input
  u(t) = ceil(4*rand(1));
  % Select a motion
  thresh = rand(1);
  new_x = find(cumsum(squeeze(mot_mod(:,:,u(t)))*x(:,t))>thresh,1);
  % Move vehicle
  x(new_x,t+1) = 1;
  % Take measurement
  thresh = rand(1);
  new_y = find(cumsum(meas_mod(:,:)*x(:,t+1))>thresh,1);
  y(new_y,t) = 1;
  % Store for plotting
```

```
...
```

- Bayes Filter:

```
...

%% Bayesian Estimation
% Prediction update
belp = squeeze(mot_mod(:,:,u(t)))*bel;

% Measurement update
bel = meas_mod(new_y,:)'.*belp;
bel = bel/norm(bel);

[pmax y_bel(t)] = max(bel);

%% Plot beliefs
...
```

Kalman Filter

- Kalman convinced NASA to run on Apollo navigation computer.

Kalman Filter Modeling Assumption

- Continuous state, inputs, measurements
- **Prior over the state** is **Gaussian**
  - $p(x_0) \sim \mathcal{N}(\mu_0, \Sigma_0)$
- **Motion model**: linear with additive Gaussian disturbances
  - $x_t = A_t x_{t-1} + B_t u_t + \epsilon_t \qquad \epsilon_t \sim \mathcal{N}(0, R_t)$
- Often, robotics systems are more easily described in continuous domain
  - Convert to discrete time using matrix exponential
  - Matlab contains tools to perform this conversion (c2d, d2c)
- **Measurement Model**: also linear with additive Gaussian noise
  - $y_t = C_t x_t + \delta_t \qquad \delta_t \sim \mathcal{N}(0, Q_t)$
  - Can add in input dependence to match up with controls literature:
    - $y_t = C_t x_t + D_t u_t + \delta_t$

Full Model Formulation:

- **State prior:**
  - $p(x_0) \sim \mathcal{N}(\mu_0, \Sigma_0)$
- **Motion model**:
  - $x_t = A_t x_{t-1} + B_t u_t + \epsilon_t \qquad \epsilon_t \sim \mathcal{N}(0, R_t)$
- **Measurement model**:
  - $y_t = C_t x_t + \delta_t \qquad \delta_t \sim \mathcal{N}(0, Q_t)$

Belief is Gaussian:

- Assume **belief is Gaussian** at time t
  - $bel(x_t) \sim \mathcal{N}(\mu_t, \Sigma_t)$
  - μ_t is the best estimate of the current state at time t
  - Σ_t is the covariance, indicating the certainty in the current estimate
  - => the predicted belif at the next time step is also **Gaussian**
    - $\overline{bel}(x_{t+1}) \sim \mathcal{N}(\overline{\mu}_{t+1}, \overline{\Sigma}_{t+1})$
  - => the belief at next time step is also **Gaussian**
    - $bel(x_{t+1}) \sim \mathcal{N}(\mu_{t+1}, \Sigma_{t+1})$

Goal:

- To find belief over state as accurately as possible given all available information
  - Minimize the mean square error of the estimate (**MMSE estimator**)

    $$\min E[(\mu_t - x_t)^2]$$

    - Same as least square problem
  - $\because$ Using an unbiased estimator:

    $$E[\mu_t - x_t] = 0$$

    - on average, your estimate is correct!
  - $\therefore$ MMSE becomes:
    - $\min E[(\mu_t - x_t)^2] = \min E[(x_t - \mu_t)^T (x_t - \mu_t)]$
    - $\equiv$ to minimizing **the trace of the error covariance matrix**:
      - => $\min tr(\Sigma_t)$

[KF Algorithm Abstract]:

- (See derivations from slides)
- At each time step t, update both sets of beliefs:

1. Prediction Update:

- Find update rule for mean, covariance of predicted belief, given input and motion model

- $\overline{\mu}_t = A_t \mu_{t-1} + B_t \mu_t$ => mean: noise in the motion is gone
- $\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$ => R_t motion model noise variance = σ_t^2

2. Measurement Update:

- Solve MMSE optimization problem to find update rule for mean, covariance of belief given measurement model and measurement

- $K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + Q_t)^{-1}$ <= **Kalman Gain**, Blending factor between prediction and measurement
- $\mu_t = \overline{\mu}_t + K_t (y_t - C_t \overline{\mu}_t)$
- $\Sigma_t = (I - K_t C_t) \overline{\Sigma}_t$

[Example]

- Temperature control
  - State is current temperature difference with outside
  - One dimensional example
  - Prior:
    - fairly certain of current temperature difference
      - $\mu_0 = 10$
      - $\Sigma_0 = 1$
  - Motion Model:
    - Decaying temperature + furnace input + disturbances (opening doors, outside effects)
      - $dt = 0.1$
      - $x_t = 0.8 x_{t-1} + 3 u_t + r_t$
      - $A = 0.8, B = 3$
      - $r_t \sim \mathcal{N}(0, 2)$
  - Measurement Model:
    - Directly measure the current temperature difference
      - $y(t) = x(t) + \delta_t$
      - $\delta_t \sim \mathcal{N}(0, 4)$
  - Controller design:
    - Bang bang control, based on current estimate of temperature difference
    - $u(t) = \begin{cases} 1 & \mu_t < 2 \\ 0 & \mu_t > 2 \\ u(t-1) & otherwise \end{cases}$

- **Simulation:**

```matlab
for t=1:length(T)

  % Select control action
  if (t>1) u(t)=u(t-1); end

  if (mu > 10)
    u(t) = 0;
  elseif (mu < 2);
    u(t) = 1;
  end

  % Update state
  e = sqrt(R)*randn(1);
  x(t+1) = A*x(t)+ B*u(t) + e;

  % Determine measurement
  d = sqrt(Q)*randn(1);
  y(t) = C*x(t+1) + d;
```

- **Estimation:**

```matlab
% Prediction update
mup = A*mu + B*u(t);
Sp = A*S*A' + R;

% Measurement update
K = Sp*C'*inv(C*Sp*C'+Q);
mu = mup + K*(y(t)-C*mup);
S = (1-K*C)*Sp;
```

  - Matrix inverse $0(n^{2.4})$, matrix multiplication O(n^{2})
  - When implementing in Matlab, inv( ) performs matrix inverse for you
  - For embeddded code, many libraries exist
    - Try Gnu Scientific Library, easy starting point

[Summary]

Summary:

- Follows same framework as **Bayes filter**
- Requires **linear motion** and **Gaussian disturbance**
- Requires **linear measurement** and **Gaussian noise**
- It is sufficient to update **mean** and **covariance** of **beliefs**, because they **remain Gaussian**
- **Prediction step** involves **addition of Gaussians**
- **Measurement step** seeks to **minimize** mean square error of **the estimate** (MMSE)
  - Expand out **covariance** from definition and measurement model
  - Assume form of **estimator**, **linear combination** of **prediction** and **measurement**
  - **Solve MMSE problem** to find **optimal linear combination**
  - **Simplify covariance** update once **gain is found**

Relation to Bayes Filter:

Relation | Problem Formulation:

- Refers to Bayes Filter | Problem Formulation
- **State prior:**
  - $bel(x_0) = p(x_0) = \mathcal{N}(\mu_0, \Sigma_0)$
- **Motion model:**
  - $p(x_t|x_{t-1}, u_t) = \mathcal{N}(A_t x_{t-1} + B_t u_t, A_t \Sigma_{t-1} A_t^T + R_t)$
- **Measurement model:**
  - $p(y_t|x_t) = \mathcal{N}(C_t x_t, Q_t)$
- **Beliefs:**
  - $bel(x_t) = \mathcal{N}(\mu_t, \Sigma_t), \qquad \overline{bel}(x_t) = \mathcal{N}(\overline{\mu}_t, \overline{\Sigma}_t)$

Relation | Algorithm:

- Refers to Bayes Filter | Algorithm

1. Prediction update (Total Probability):

- Insert normal distributions:

$$\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$$
$$= \eta \int e^{-1/2(x_t - A_t x_{t-1} - B_t u_t)^T \; R_T^{-1}(x_t - A_t x_{t-1} - B_t u_t)} e^{-1/2(x_{t-1}-\mu_{t-1})^T \Sigma_{t-1}^{-1}(x_{t-1}-\mu_{t-1})} dx_{t-1}$$

- Separate out terms that depend on current state
- Manipulate remaining integral into a **Gaussian PDF** form of previous state
- Integrate over full range => to get 1
- Manipulate remaining terms and solve for Kalman prediction equations:

  $$\sim \mathcal{N}(A_t \mu_{t-1} + B_t u_t, A_t \Sigma_{t-1} A_t^T + R_t)$$

2. Measurement Update (Bayes Theorem):

- Measurement update:

$$bel(x_t) = \eta p(y_t|x_t) \overline{bel}(x_t)$$
$$= \eta e^{-1/2(y_t - C_t x_t)^T Q_t^{-1}(y_t - C_t x_t)} e^{-1/2(x_t - \bar{\mu}_t)^T \overline{\Sigma}_t^{-1}(x_t - \bar{\mu}_t)}$$

- Reorganize exponents and note it **remains a Gaussian**
- For any Gaussian:
  - Second derivative of exponent is inverse of covariance
  - Mean minimizes exponent
    - Set first derivative of exponent to 0 and solve
- Use this to solve for mean and covariance of belief

  $$= \mathcal{N}(\bar{\mu}_t + K_t(y_t - C_t \bar{\mu}_t), (I - K_t C_t)\overline{\Sigma}_t)$$

  - where K_t is the Kalman gain

Variable Summary:

- Belief mean is tradeoff between prediction and measurement
  - Kalman gain determines how to blend estimates
- If Q_t is large, inverse is small => so Kalman gain remains small
  - When measurements are high in covariance, don't trust them!
- If R_t is large, then so is predicted belief covariance, => so Kalman gain becomes large
  - When model is affected by large unkown disturbances, don;t truct the predicted motion!!!

Extended Kalman Filter

- A direct generalization of the Kalman filter to nonlinear motion and measurement models
  - Relies on linearization about current estimate
  - Works well when the problem **maintains locally linear and Gaussian characteristics**
  - Computationally similar to Kalman Filter
  - Covariance can diverge when approximation is poor!

EKF | Modeling Assumption:

- **Prior** over the state is Gaussian

  $$p(x_0) \sim \mathcal{N}(\mu_0, \Sigma_0)$$

- **Motion mode**, **nonlinear** but still with additive Gaussian disturbances

  $$x_t = g(x_{t-1}, u_t) + \epsilon_t \qquad \epsilon_t \sim \mathcal{N}(0, R_t)$$

- **Measurement model**, also **nonlinear** with additive Gaussing noise

  $$y_t = h(x_t) + \delta_t \qquad \delta_t \sim \mathcal{N}(0, Q_t)$$

- Non-linearity destroys certainty that beliefs remain Gaussian

[EKF Algorithm Abstract]

0. Linearization with First Order Taylor Series Expansion

- Only valid near point of linearization
- Motion model:

$$
\begin{aligned}
g(x_{t-1}, u_t) &\approx g(\mu_{t-1}, u_t) + \frac{\partial}{\partial x_{t-1}} g(x_{t-1}, u_t)\Big|_{x_{t-1}=\mu_{t-1}} \cdot (x_{t-1} - \mu_{t-1}) \\
&= g(\mu_{t-1}, u_t) + G_t \cdot (x_{t-1} - \mu_{t-1})
\end{aligned}
$$

- Measurement model:

$$
\begin{aligned}
h(x_t) &\approx h(\overline{\mu}_t) + \frac{\partial}{\partial x_t} h(x_t)\Big|_{x_t=\mu_t} \cdot (x_t - \overline{\mu}_t) \\
&= h(\overline{\mu}_t, u_t) + H_t \cdot (x_t - \overline{\mu}_t)
\end{aligned}
$$

1. Prediction Update

- $G_t = \frac{\partial}{\partial x_{t-1}} g(x_{t-1}, u_t)\Big|_{x_{t-1}=\mu_{t-1}}$
- $\overline{\mu}_t = g(\mu_{t-1}, u_t)$
- $\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$

2. Measurement Update

- $H_t = \frac{\partial}{\partial x_t} h(x_t)\Big|_{x_t=\mu_t}$
- $K_t = \overline{\Sigma}_t H_t^T (H_t \overline{\Sigma}_t H_t^T + Q_t)^{-1}$
- $\mu_t = \overline{\mu}_t + K_t(y_t - h(\overline{\mu}_t))$
- $\Sigma_t = (I - K_t H_t)\overline{\Sigma}_t$

Sample Code:

```
%% Extended Kalman Filter Estimation % Prediction update
mup = Ad*mu;
Sp = Ad*S*Ad' + R;
% Measurement update
Ht = [(mup(1))/(sqrt(mup(1)^2 + mup(3)^2));
0; (mup(3))/(sqrt(mup(1)^2 + mup(3)^2))]’;
K = Sp*Ht'*inv(Ht*Sp*Ht'+Q);
mu = mup + K*(y(:,t)-sqrt(mup(1)^2 + mup(3)^2));
S = (eye(n)-K*Ht)*Sp;
```

Summary

- Direct extension of KF to nonlinear models
- Use Taylor series expansion to find locally linear approximations
- No longer optimal
- Most effective when covariance is low
    - Local linear approximation more likely to be accurate over range of distribution
- Covariance update may diverge
- The EKF used linearization about the predicted/previous state estimate to update the mean and covariance of the current estimate
    - Approximation of a nonlinear transformation of a Gaussian distribution by linear transformation of the mean and covariance

[Summary | Comparison Table]

<table>
<tr><th></th><th>Bayes Filter</th><th>Kalman Filter</th><th>Extended Kalman Filter</th></tr>
<tr><td>Prior State</td><td>$bel(x_0) = p(x_0)$</td><td>$= \mathcal{N}(\mu_0, \Sigma_0)$</td><td>$p(x_0) \sim \mathcal{N}(\mu_0, \Sigma_0)$</td></tr>
<tr><td>Motion Model</td><td>$p(x_t \mid x_{t-1}, u_t)$</td><td>$= \mathcal{N}(A_t x_{t-1} + B_t u_t, A_t \Sigma_{t-1} A_t^T + R_t)$</td><td>$x_t = g(x_{t-1}, u_t) + \epsilon_t\ , \epsilon_t \sim \mathcal{N}(0, R_t)$</td></tr>
<tr><td>Measurement Model</td><td>$p(yt \mid xt)$</td><td>$= \mathcal{N}(C_t x_t, Q_t)$</td><td>$y_t = h(x_t) + \delta_t\ , \delta_t \sim \mathcal{N}(0, Q_t)$</td></tr>
<tr><td>Beliefs</td><td>$bel(x_t) = p(x_t \mid y_{1:t}, u_{1:t})$</td><td>$bel(x_t) = \mathcal{N}(\mu_t, \Sigma_t), \qquad \overline{bel}(x_t) = \mathcal{N}(\overline{\mu}_t, \overline{\Sigma}_t)$</td><td>Remain Gaussian</td></tr>
<tr><td>1. Prediction Update</td><td>$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$</td><td>$= \eta \int \Big[e^{-1/2(x_t - A_t x_{t-1} - B_t u_t)^T\ R_T^{-1} (x_t - A_t x_{t-1} - B_t u_t)}$
$e^{-1/2(x_{t-1} - \mu_{t-1})^T \Sigma_{t-1}^{-1} (x_{t-1} - \mu_{t-1})} \Big]\ dx_{t-1}$</td><td></td></tr>
<tr><td>(at time t)</td><td>$\overline{bel}(x_t) = \sum p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$</td><td>- $\overline{\mu}_t = A_t \mu_{t-1} + B_t \mu_t$
- $\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$</td><td>$G_t = \frac{\partial}{\partial x_{t-1}} g(x_{t-1}, u_t)\Big|_{x_{t-1} = \mu_{t-1}}\ \overline{\mu}_t = g(\mu_{t-1}, u_t)\ \overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$</td></tr>
<tr><td>2. Measurement Update</td><td>$bel(x_t) = \eta\, p(y_t \mid x_t) \overline{bel}(x_t)$</td><td>$= \eta e^{-1/2(y_t - C_t x_t)^T Q_t^{-1} (y_t - C_t x_t)}\, e^{-1/2(x_t - \bar{\mu}_t)^T \overline{\Sigma}_t^{-1} (x_t - \bar{\mu}_t)}$</td><td></td></tr>
<tr><td>(at time t)</td><td>$bel(x_t) = \eta\, p(y_t \mid x_t) \overline{bel}(x_t)$</td><td>- $K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + Q_t)^{-1}$
- $\mu_t = \overline{\mu}_t + K_t (y_t - C_t \overline{\mu}_t)$
- $\Sigma_t = (I - K_t C_t) \overline{\Sigma}_t$</td><td>$H_t = \frac{\partial}{\partial x_t} h(x_t)\Big|_{x_t = \mu_t}\ K_t = \overline{\Sigma}_t H_t^T (H_t \overline{\Sigma}_t H_t^T + Q_t)^{-1}$
$\mu_t = \overline{\mu}_t + K_t (y_t - h(\overline{\mu}_t))\ \Sigma_t = (I - K_t H_t) \overline{\Sigma}_t$</td></tr>
<tr><td>Summary</td><td>- Bayes Filter | Algorithm
- Bayes Filter | Problem Formulation

- If state, measurements, inputs are DISCRETE,
-- can directly implement Bayes Filter
--- Prediction update is summation over discrete states
--- Measurement update is multiplication of two vectors

- Else, if they are CONTINUOUS
-- must define model or approximation to enable computation:
---- KF (Kalman Filter)
-------- Linear motion models
-------- Linear measurement models
-------- Additive Gaussian disturbance and noise distributions
---- EKF / UKF (Extended Kalman Filter / Unscented Kalman Filter)
-------- Non-linear motion models
-------- Non-linear measurement models
-------- Additive Gaussian disturbance and noise distributions
---- PF (Particle Filter)
-------- (Dis)continuous motion models
-------- (Dis)continuous measurement models
-------- General disturbance and noise distributions</td><td>- KF Algorithm Abstract

- Follows same framework as Bayes filter
- Requires linear motion and Gaussian disturbance
- Requires linear measurement and Gaussian noise
- It is sufficient to update mean and covariance of beliefs, because they remain Gaussian**Prediction step** involves addition of Gaussians
- Measurement step seeks to minimize mean square error of the estimate (MMSE)
- - Expand out covariance from definition and measurement model
- - Assume form of estimator, linear combination of prediction and measurement
- - Solve MMSE problem to find optimal linear combination
- - Simplify covariance update once gain is found</td><td>- EKF Algorithm Abstract

- non-linear measurements and motion models
- no longer optimal
- Most effective when covariance is low
- covariance update may diverge
- Approximation of a nonlinear transformation of a Gaussian distribution by linear tranformation of the mean and covariance</td></tr>
</table>