Lecture 17

Object Tracking

ECE 1390/2390

Kalman FIlter

- Hierarchical Noise Models

Observation layer

Measurement (Lead field) model

Measurement noise

Observations

$$Y(t) = F(X(t), U(t)) + \epsilon(t)$$

Control/Input vector [optional]

State/process noise

Hidden-state layer

States

$$X(t) = H(X(t - \delta t)), U'(t)) + v(t)$$

State update model

- Hierarchical Noise Models

Observation layer

Measurement (Lead field) model

Measurement noise

Observations

$$Y(t) = F(X(t), U(t)) + \epsilon(t)$$

Control/Input vector [optional]

State/process noise

Hidden-state layer I

States

$$X(t) = H(Z(t - \delta t)), U'(t)) + v(t)$$

State update model

Hidden-state layer II

$$Z(t) = K(Z(t - \delta t)), U''(t)) + w(t)$$

Example

Observation Layer

$$\begin{bmatrix} Y_{lat}[n] \\ Y_{long}[n] \end{bmatrix} = \begin{bmatrix} X_{lat}[n] \\ X_{long}[n] \end{bmatrix} + \varepsilon$$

Position Layer

$$\begin{bmatrix} X_{lat}[n] \\ X_{long}[n] \end{bmatrix} = \begin{bmatrix} X_{lat}[n-1] \\ X_{long}[n-1] \end{bmatrix} + \mathrm{T} \cdot \begin{bmatrix} \cos(\theta[n]) & \sin(\theta[n]) \\ -\sin(\theta[n]) & \cos(\theta[n]) \end{bmatrix} \cdot V_{elocity}[n] + p$$

Velocity Layer

$$\begin{bmatrix} V_{elocity}[n] \\ \theta[n] \end{bmatrix} = \begin{bmatrix} V_{elocity}[n-1] \\ \theta[n-1] \end{bmatrix} + \mathrm{T} \cdot \begin{bmatrix} A_{cc}[n] \\ W_{acc}[n] \end{bmatrix} + v$$

Acceleration Layer

$$\begin{bmatrix} A_{cc}[n] \\ W_{acc}[n] \end{bmatrix} = \begin{bmatrix} A_{cc}[n-1] \\ W_{acc}[n-1] \end{bmatrix} + \omega$$

Example

$$Y[n] = A \cdot X[n] + B \cdot U[n] + e$$

$$\begin{bmatrix} Ylat(n) \\ Ylong(n) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} Xlat(n) \\ Xlong(n) \\ vel(n) \\ \theta(n) \\ Acc(n) \\ \omega(n) \end{bmatrix}$$

$$\mathrm{X}[n] = C \cdot X[n-1] + D \cdot U[n] + v$$

$$\begin{bmatrix} Xlat(n) \\ Xlong(n) \\ vel(n) \\ \theta(n) \\ Acc(n) \\ \omega(n) \end{bmatrix} = \begin{bmatrix} 1 & 0 & T * \cos(\theta(n-1)) & T * sin(\theta(n-1)) & 0 & 0 \\ 0 & 0 & -T * \sin(\theta(n-1)) & T * \cos(\theta(n-1)) & 0 & 0 \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Xlat(n-1) \\ Xlong(n-1) \\ vel(n-1) \\ \theta(n-1) \\ Acc(n-1) \\ \omega(n-1) \end{bmatrix}$$

$$Y[n] = A \cdot X[n] + B \cdot U[n] + e \qquad \mathrm{X}[n] = C \cdot X[n-1] + D \cdot U[n] + v$$

$$\hat{X}_{pred}[n] = \mathrm{C} \cdot \mathrm{X}[\mathrm{n}\text{-}1] + D \cdot U[n]$$

$$\hat{Y}_{pred}[n] = A \cdot \hat{X}_{pred}[n] + B \cdot U[n]$$

$$\hat{P}_{pred}[n] = C \cdot P[n] \cdot C^T + Q$$

Prediction

$$err[n] = Y_{obs}[n] - \hat{Y}_{pred}[n]$$

$$S[n] = A \cdot \hat{P}_{pred}[n] \cdot A^T + R$$

$$K_{gain} = \hat{P}_{pred}[n] \cdot A^T \cdot S[n]^{-1}$$

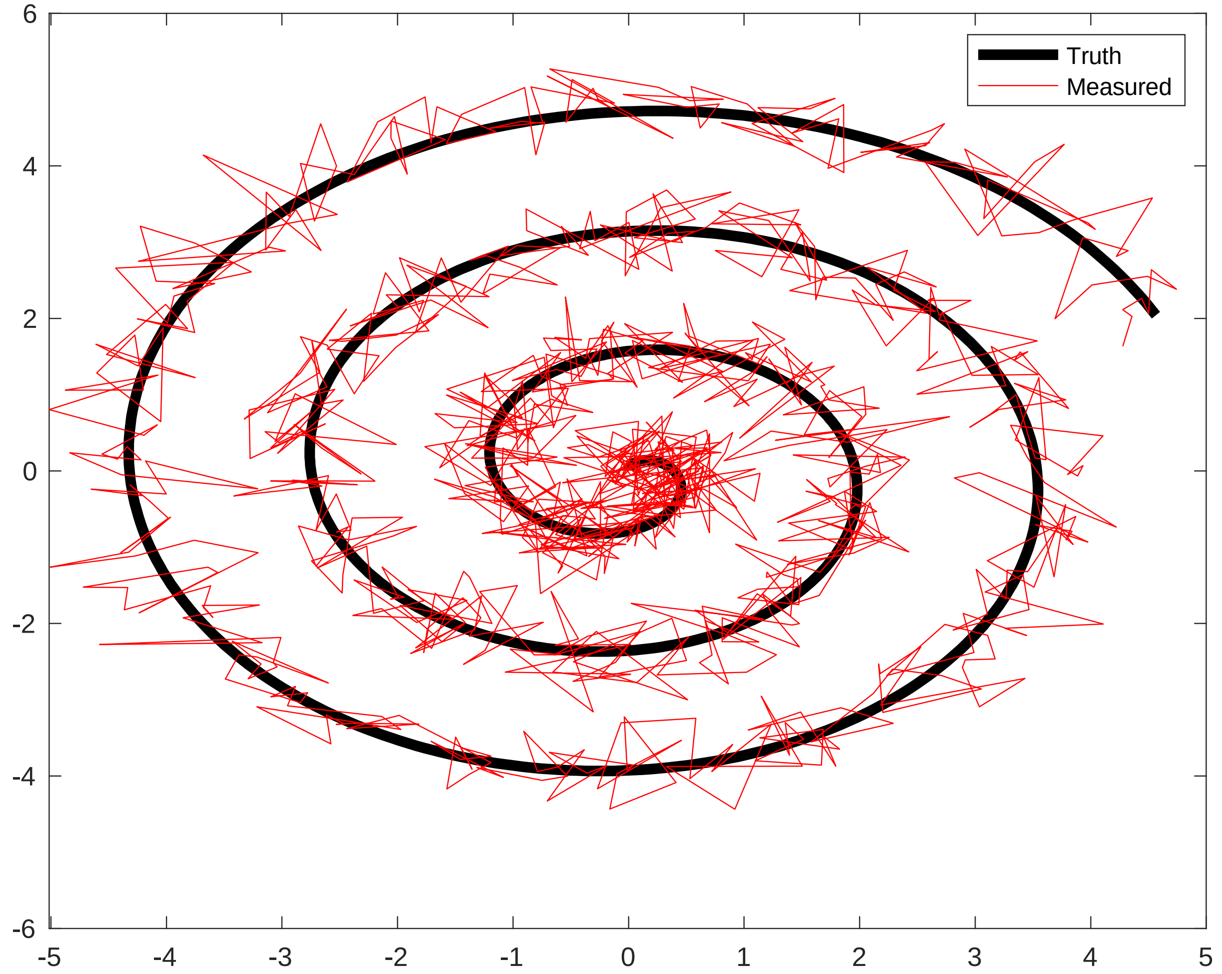
Update

$$X[n] = K_{gain} \cdot err[n] + \hat{X}_{pred}[n]$$

$$P[n] = \left(I - K_{gain} \cdot A\right) \cdot \hat{P}_{pred}[n]$$

Correction

```
>> A=eye(2);
>> nk=1000;
>> xtrue = [[1:nk]./200.*sin([1:nk]/50);...
            [1:nk]./200.*cos([1:nk]/50)];
>> y=A*xtrue+randn(2,nk)/3;
```




```
R=.1*eye(2);
Q=.2*eye(2);

x=zeros(2,nk);
P=eye(2);
for k=2:nk
        xhat=C*x(:,k-1);
        yhat = A*xhat;
        Phat = C*P*C'+Q;
        err = y(:,k)-yhat;
        S = A*Phat*A'+R;
        K = Phat*A'*inv(S);
        x(:,k)=xhat+K*err;
        P=(eye(2)-K*A)*Phat;
end
```

Optical Flow

$$I(u, v, t) \rightarrow I(u + \Delta u, v + \Delta v, t + \Delta t)$$

Taylor expansion

$$I(u + \Delta u, v + \Delta v, t + \Delta t) \approx I(u, v, t) + \frac{dI}{du}\delta u + \frac{dI}{dv}\delta \mathrm{v} + \frac{dI}{dt}\delta t$$

Assume that intensity doesn't change (just moves around)

$$I(u, v, t) = I(u + \Delta u, v + \Delta v, t + \Delta t)$$

$$0 = \frac{dI}{du}\delta u + \frac{dI}{dv}\delta \mathrm{v} + \frac{dI}{dt}\delta t$$

Optical Flow

Unknowns (du/dt & dv/dt)

$$-\frac{dI}{dt} = \frac{dI}{du}\frac{du}{dt} + \frac{dI}{dv}\frac{dv}{dt}$$

Change in intensity at pixel

Gradient in horizontal (u)

Gradient in vertical (v)

Each pixel has one equation and two unknowns

- Lucas-Kanade Method (Sparse flow)
- Gunnar Farneback Method (dense flow)

Optical Flow

Lucas-Kanade Method (Sparse flow)

- Track only corner points
- Find Shi-Tomasi points between frames (cv2.goodPointsToTrack(.))

```
# params for corner detection
feature_params = dict( maxCorners = 100, qualityLevel = 0.3, minDistance = 7, blockSize = 7 )

# Parameters for lucas kanade optical flow
lk_params = dict( winSize = (15, 15), maxLevel = 2, criteria = (cv2.TERM_CRITERIA_EPS
        |cv2.TERM_CRITERIA_COUNT,10, 0.03))

old_pts = cv2.goodFeaturesToTrack(old_gray, mask = None, **feature_params)

New_pts, status, err = cv2.calcOpticalFlowPyrLK(old_gray, new_gray, old_pts, None, **lk_params)

# Select good points
good_new = new_pts[status == 1]
good_old = old_pts[status == 1]
```

Optical Flow

Lucas-Kanade Method (Sparse flow)

Optical Flow

$$-\frac{dI}{dt} = \frac{dI}{du}\frac{du}{dt} + \frac{dI}{dv}\frac{dv}{dt}$$

- Gunnar Farneback Method (dense flow)

Make image pyramid

Polynomial Approximation

$$\tilde{I}(u, v, t = 1) = \sum_{j=-2}^{2} \sum_{i=-2}^{2} \alpha_{i,j} * I(u + i, v + j, t = 0)$$

Iterative estimate du & dv

$$I(u, v, t = 0) - I(u, v, t = 1) = \frac{d\tilde{I}(u,v)}{du}\frac{du}{dt} + \frac{d\tilde{I}(u,v)}{dv}\frac{dv}{dt}$$

$$\alpha_{i,j} = \begin{cases} 1 & i, j = du, dv \\ 0 & else \end{cases}$$

Optical Flow

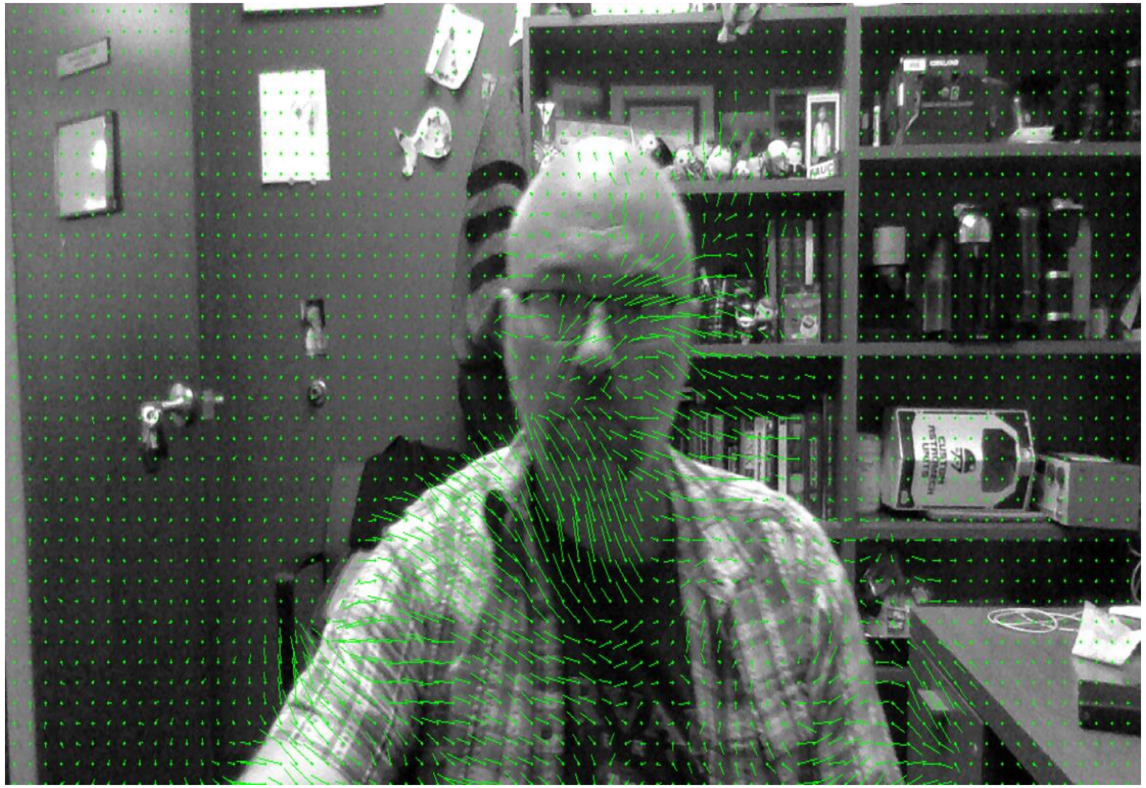
- Gunnar Farneback Method (dense flow)

```
flow = cv2.calcOpticalFlowFarneback(old_gray_img, new_gray_img,
        flow=None,
        pyr_scale=0.5, # Scale for image pyramid (0.5 -> halves image for each level)
        levels=3, # number of levels in the pyramid
        winsize=15,# larger the window, the less sensitive to noise, but the more blurred
        iterations=3,
        poly_n=5, # Number of neighbors to build polynomial expansion (usually 5 or 7)
        poly_sigma=1.1). # smoothness of polynomial (usually 1.1 for poly=5 & 1.5 for poly 7)

Flow is a <n x m x 2> image and is the du/dt and dv/dt for each pixel location
```

Optical Flow

- Gunnar Farneback Method (dense flow)



Single Object tracking

- MeanShift

Center of mass of intensities inside circle

Center of mass of intensities inside (larger) circle

Size of region-of-interest in current and next frame are fixed

Single Object tracking

- CAMShift method (continuous adaptive mean shift)

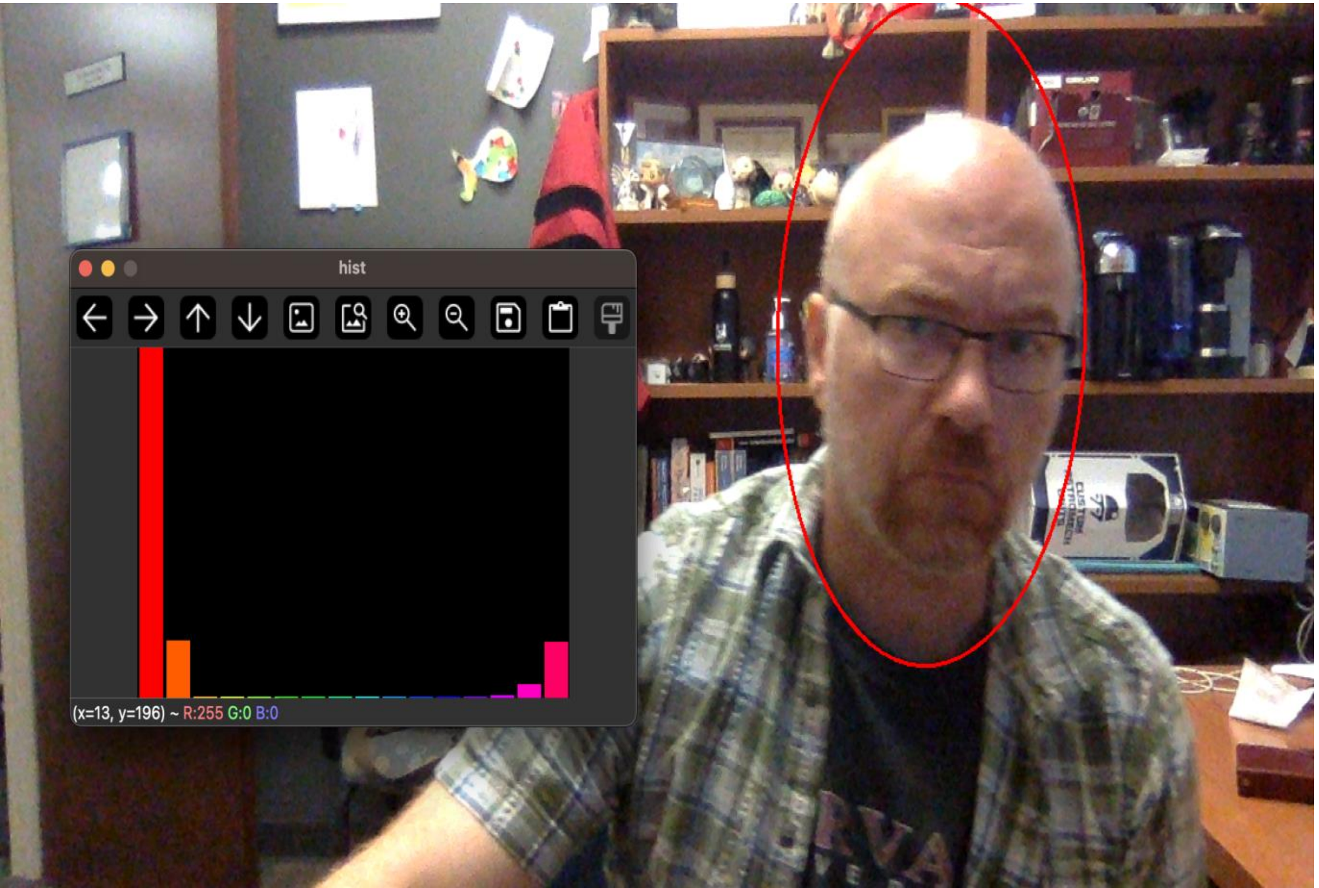
Center of mass of intensities inside circle

Center of mass of intensities inside (larger) circle

Size of region-of-interest is also adjusted based on covariance of intensities

Single Object tracking

- CAMShift method (continuous adaptive mean shift)



Boosting tracker

- Similar to Haar cascade model
- Requires training at runtime with positive/negative data
- User labels object to be tracked (or by some detection method)
- Given new frame, run classifier on all nearby pixels to previous known location
- Update positives/negatives and training as we add frames

MIL (multiple instance learning)

- Similar to Boosting tracker, but uses multiple samples around the point to define positive “bags” (collections of images related to the positive image)

- Less prone to drift cf Boosting
- Better job at partial occlusions than Boost but cannot track over full occlusions

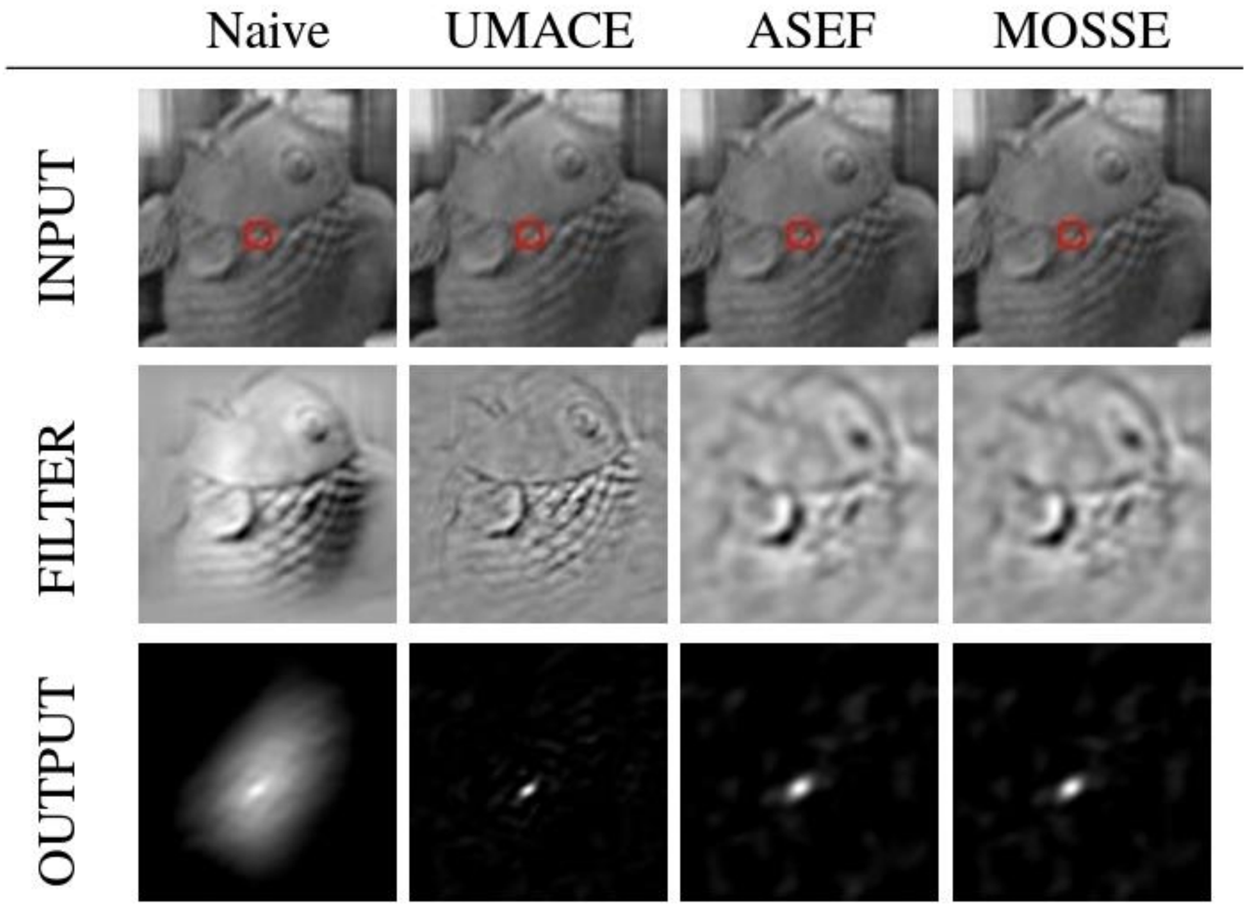
KCF (kernelized correlation function) tracker
CSRT (discriminate correlation filter)
MOSSE (minimum output sum of squared error

1) Given a test texture to track (h)
2) Compute FFT of texture (H)
3) Compute correlation of h with new image (f)

$$g = f \otimes h$$

$$\mathrm{G} = \mathrm{F} \cdot \mathrm{H}^*$$

find max of correlation



Visual Object Tracking using Adaptive Correlation Filters

David S. Bolme J. Ross Beveridge Bruce A. Draper Yui Man Lui


frame
tracker state
PSR: 47.88

Lecture Notes on Kalman filters

Example

Observation Layer

$$\begin{bmatrix} Y_{lat}[n] \\ Y_{long}[n] \end{bmatrix} = \begin{bmatrix} X_{lat}[n] \\ X_{long}[n] \end{bmatrix} + \varepsilon$$

Position Layer

$$\begin{bmatrix} X_{lat}[n] \\ X_{long}[n] \end{bmatrix} = \begin{bmatrix} X_{lat}[n-1] \\ X_{long}[n-1] \end{bmatrix} + p$$

$$Y[n] = X[n] + e$$

$$\mathrm{X}[n] = X[n-1] + p$$

$$Y[n] = A \cdot X[n] + B \cdot U[n] + e$$

$$\mathrm{X}[n] = C \cdot X[n-1] + D \cdot U[n] + v$$

Example

Observation Layer

$$Y[n] = X[n] + e$$

$$e = N(0, R)$$

Position Layer

$$\mathrm{X}[n] = X[n-1] + v$$

$$v = N(0, Q)$$

Example

Given X[n-1], where do I expect the car to be at n:

$$\hat{X}_{pred}[n]=\mathrm{X[n\text{-}1]}$$

$$\hat{Y}_{pred}\,[n] = \hat{X}_{pred}[n]$$

After I measure the position, how far off was my estimate:

$$err[n] = Y_{obs}[n] - \hat{Y}_{pred}\,[n]$$

Update the estimate of X[n] between where I expected it to be and where I measured it :

$$X[n] = K_{gain} \cdot err[n] + \hat{X}_{pred}[n]$$

$$K_{gain} \sim \begin{cases} 1 & \text{Update is solely determined by Y[n]} \\ 0 & \text{Update is solely determined by X}[\mathrm{n} - 1] \end{cases}$$

Kalman Gain

$$Y = A \cdot X + e$$

Least-squares

$$\min_X \left(|Y - AX|^2\right)$$

Prior expectation of X

Weighted Least-squares

$$\min_X \left(|Y - AX|_R^2 + |X - X_0|_Q^2\right)$$

Notation denotes:

$$|\Psi|_\Omega^2 \equiv \Psi \cdot \Omega^{-1} \cdot \Psi^T$$

Kalman Gain

$$Y = A \cdot X + e$$

$$\min_X \left(|Y - AX|_R^2 + |X - X_0|_Q^2 \right)$$

$$X = (A^T \cdot R^{-1} \cdot A + Q^{-1})^{-1} A^T \cdot R^{-1} \cdot (Y - A \cdot X_0) + X_0$$

<u>Special Case (Tikhonov regularization/ Ridge regression)</u>

$X_0 = 0$ $\qquad R = \varepsilon \cdot I$ $\qquad$ Q$= v \cdot I$ $\qquad \lambda = \frac{\varepsilon}{v}$ ← 1/Signal-to-noise ratio

$$X = (A^T \cdot A + \lambda \cdot I)^{-1} A^T \cdot Y$$

Kalman Gain

$$Y = A \cdot X + e$$

$$\min_X \left(|Y - AX|_R^2 + |X - X_0|_Q^2 \right)$$

$$X = \underbrace{(A^T \cdot R^{-1} \cdot A + Q^{-1})^{-1} A^T \cdot R^{-1}} \cdot (Y - A \cdot X_0) + X_0$$

$$X[n] = K_{gain} \cdot (Y_{obs}[n] - \hat{Y}_{pred}\,[n]) + \hat{X}_{pred}[n]$$

$$X = Q \cdot A^T \cdot (A \cdot Q \cdot A^T + R)^{-1} \cdot (Y - A \cdot X_0) + X_0$$

$$Y[n] = X[n] + e \qquad \mathrm{X}[n] = X[n-1] + v$$

$$\hat{X}_{pred}[n]=\mathrm{X[n\text{-}1]}$$

$$\hat{Y}_{pred}[n] = \hat{X}_{pred}[n]$$

$$\hat{P}_{pred}[n] = P[n] + Q$$

Prediction

$$err[n] = Y_{obs}[n] - \hat{Y}_{pred}[n]$$

$$S[n] = \hat{P}_{pred}[n] + R$$

$$K_{gain} = \hat{P}_{pred}[n] \cdot S[n]^{-1}$$

Update

$$X[n] = K_{gain} \cdot err[n] + \hat{X}_{pred}[n]$$

$$P[n] = \left(I - K_{gain}\right) \cdot \hat{P}_{pred}[n]$$

Correction

$$Y[n] = A \cdot X[n] + B \cdot U[n] + e \qquad \mathrm{X}[n] = C \cdot X[n-1] + D \cdot U[n] + v$$

$$\hat{X}_{pred}[n] = \mathrm{C} \cdot \mathrm{X}[\mathrm{n}\text{-}1] + D \cdot U[n]$$

$$\hat{Y}_{pred}[n] = A \cdot \hat{X}_{pred}[n] + B \cdot U[n]$$

$$\hat{P}_{pred}[n] = C \cdot P[n] \cdot C^T + Q$$

Prediction

$$err[n] = Y_{obs}[n] - \hat{Y}_{pred}[n]$$

$$S[n] = A \cdot \hat{P}_{pred}[n] \cdot A^T + R$$

$$K_{gain} = \hat{P}_{pred}[n] \cdot A^T \cdot S[n]^{-1}$$

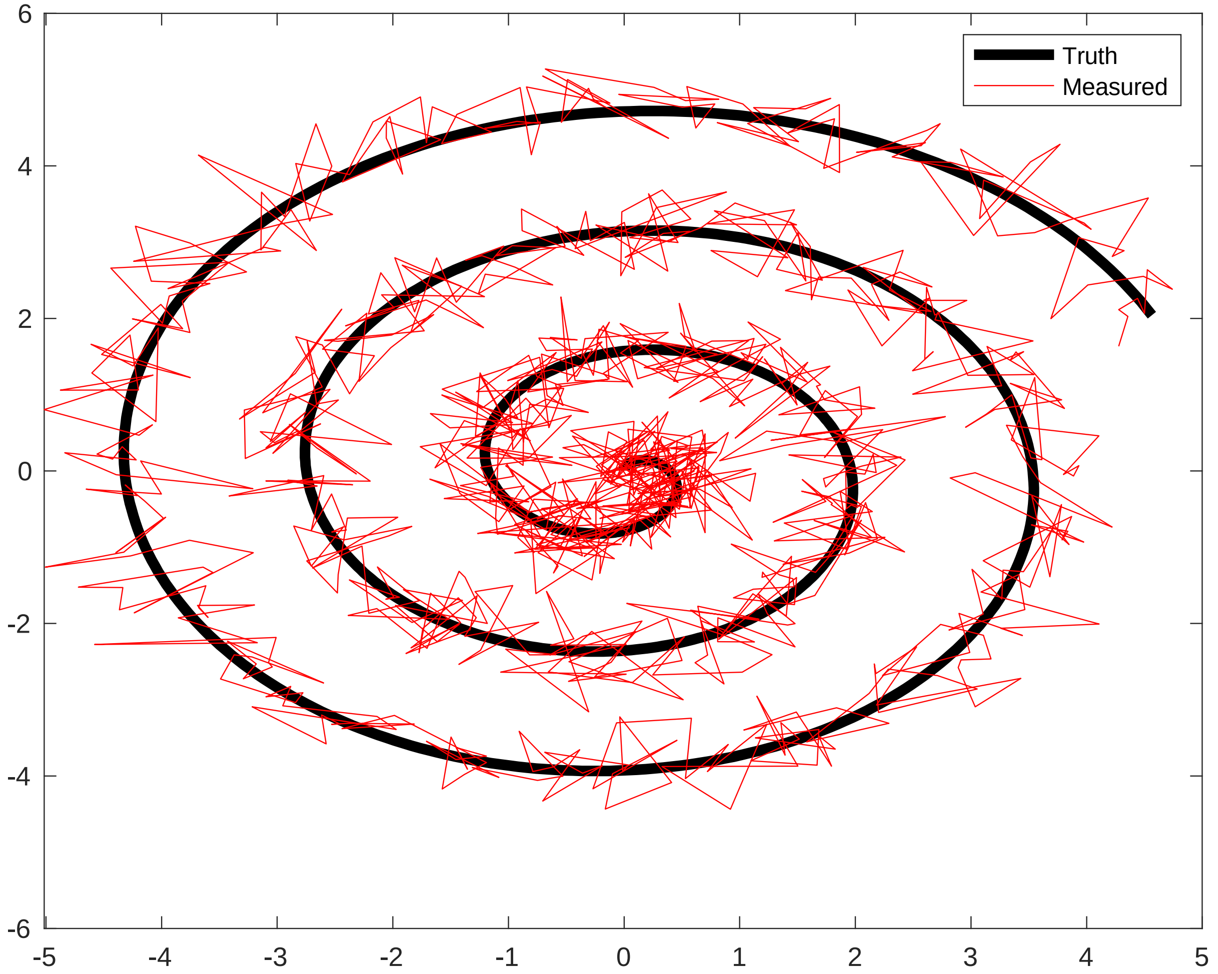
Update

$$X[n] = K_{gain} \cdot err[n] + \hat{X}_{pred}[n]$$

$$P[n] = \left(I - K_{gain} \cdot A\right) \cdot \hat{P}_{pred}[n]$$

Correction

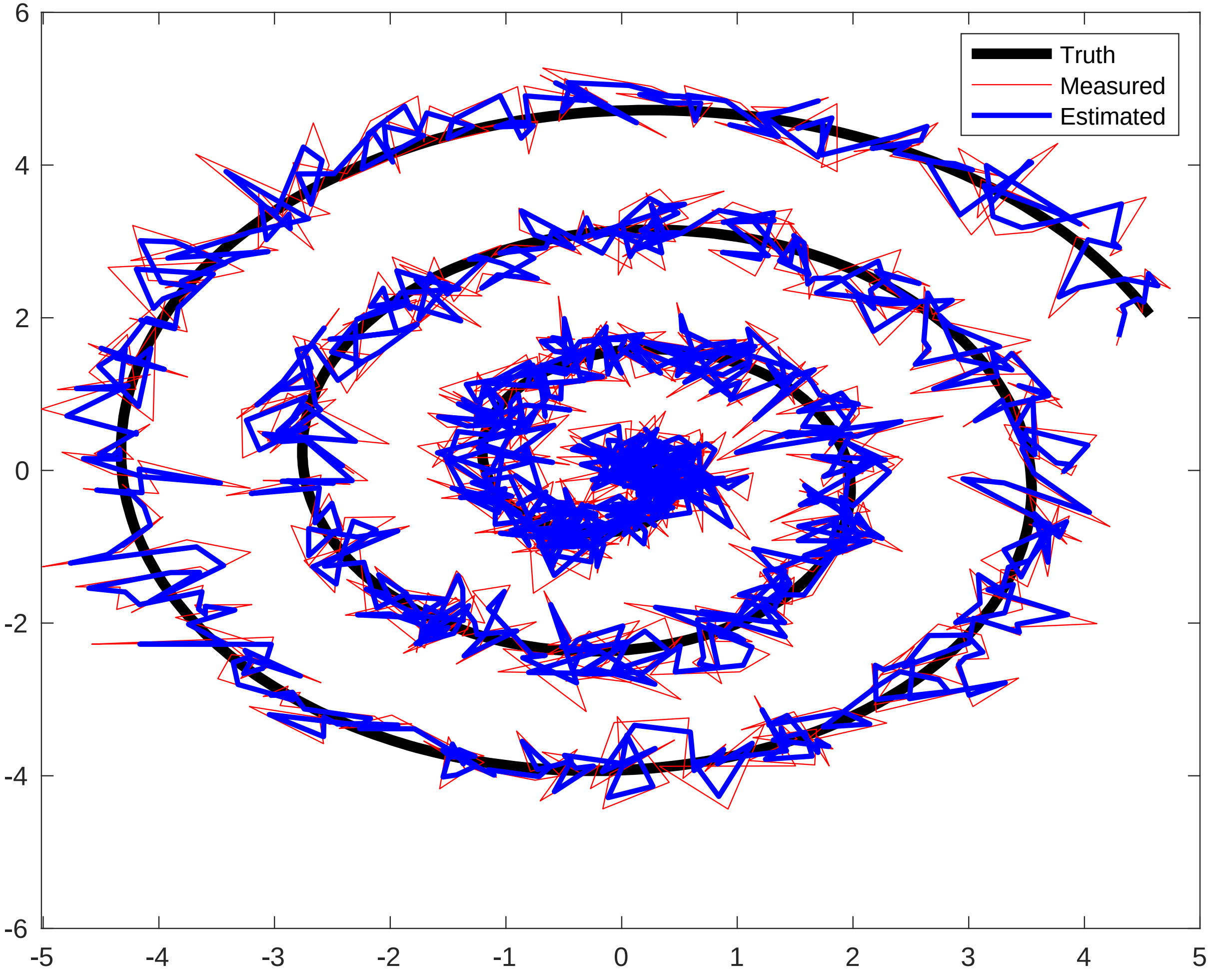
```
>> A=eye(2);
>> nk=1000;
>> xtrue = [[1:nk]./200.*sin([1:nk]/50);...
            [1:nk]./200.*cos([1:nk]/50)];
>> y=A*xtrue+randn(2,nk)/3;
```




```
R=.1*eye(2);
Q=.2*eye(2);

x=zeros(2,nk);
P=eye(2);
for k=2:nk
        xhat=C*x(:,k-1);
        yhat = A*xhat;
        Phat = C*P*C'+Q;
        err = y(:,k)-yhat;
        S = A*Phat*A'+R;
        K = Phat*A'*inv(S);
        x(:,k)=xhat+K*err;
        P=(eye(2)-K*A)*Phat;
end
```

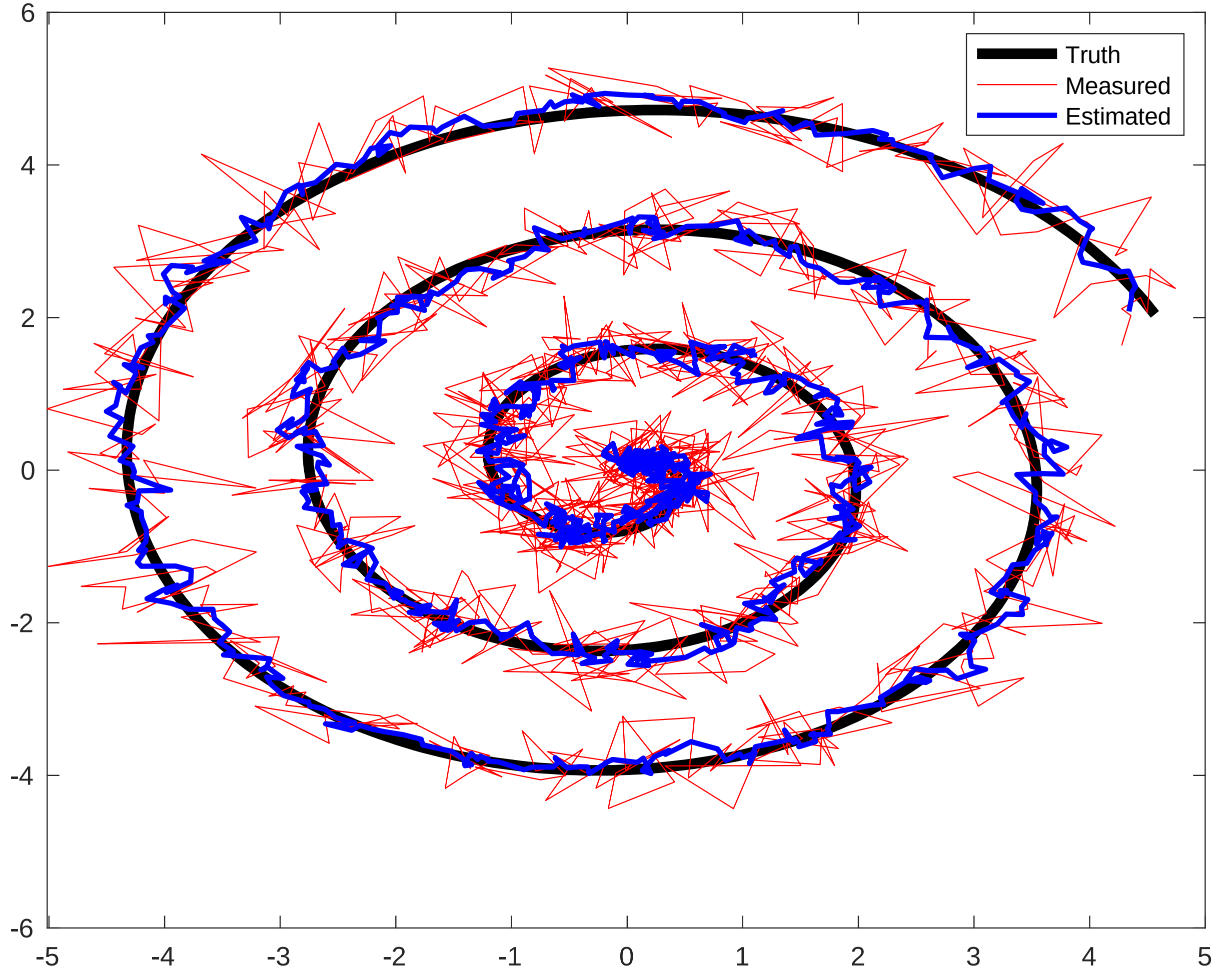
```
>> A=eye(2);
>> nk=1000;
>> xtrue = [[1:nk]./200.*sin([1:nk]/50);...
            [1:nk]./200.*cos([1:nk]/50)];
>> y=A*xtrue+randn(2,nk)/3;
```




```
R=.1*eye(2);
Q=.2*eye(2);

x=zeros(2,nk);
P=eye(2);
for k=2:nk
        xhat=C*x(:,k-1);
        yhat = A*xhat;
        Phat = C*P*C'+Q;
        err = y(:,k)-yhat;
        S = A*Phat*A'+R;
        K = Phat*A'*inv(S);
        x(:,k)=xhat+K*err;
        P=(eye(2)-K*A)*Phat;
end
```

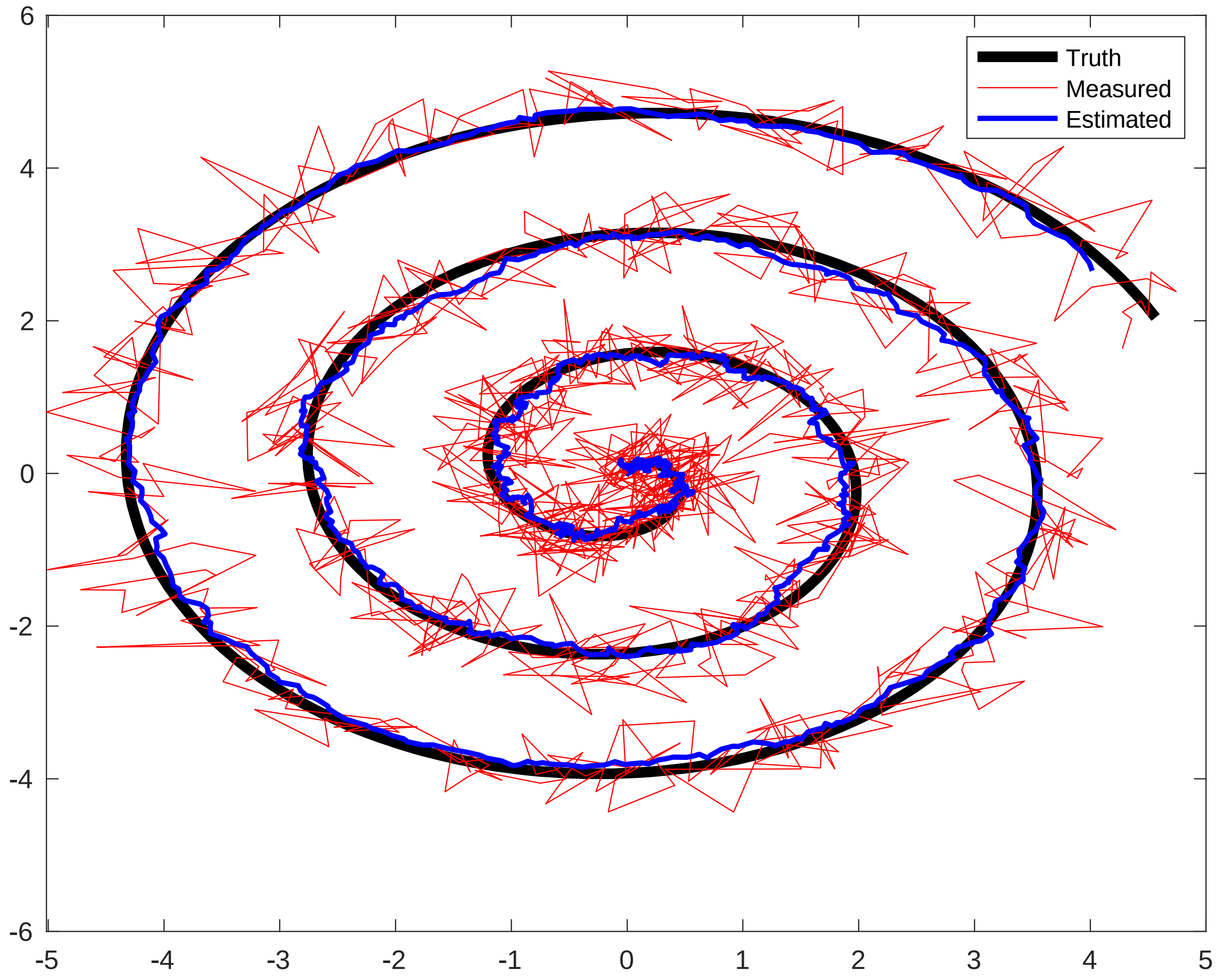
```
>> A=eye(2);
>> nk=1000;
>> xtrue = [[1:nk]./200.*sin([1:nk]/50);...
            [1:nk]./200.*cos([1:nk]/50)];
>> y=A*xtrue+randn(2,nk)/3;
```




```
R=2*eye(2);
Q=.2*eye(2);

x=zeros(2,nk);
P=eye(2);
for k=2:nk
        xhat=C*x(:,k-1);
        yhat = A*xhat;
        Phat = C*P*C'+Q;
        err = y(:,k)-yhat;
        S = A*Phat*A'+R;
        K = Phat*A'*inv(S);
        x(:,k)=xhat+K*err;
        P=(eye(2)-K*A)*Phat;
end
```

```
>> A=eye(2);
>> nk=1000;
>> xtrue = [[1:nk]./200.*sin([1:nk]/50);...
            [1:nk]./200.*cos([1:nk]/50)];
>> y=A*xtrue+randn(2,nk)/3;
```




```
R=20*eye(2);
Q=.2*eye(2);

x=zeros(2,nk);
P=eye(2);
for k=2:nk
        xhat=C*x(:,k-1);
        yhat = A*xhat;
        Phat = C*P*C'+Q;
        err = y(:,k)-yhat;
        S = A*Phat*A'+R;
        K = Phat*A'*inv(S);
        x(:,k)=xhat+K*err;
        P=(eye(2)-K*A)*Phat;
end
```

Fixed Interval Smoother

For (k=0; k<n; k++):

$$\hat{X}_{k|k-1} = C \cdot X_{k-1|k-1} + D \cdot U_k$$

$$\hat{Y}_k = A \cdot \hat{X}_{k|k-1} + B \cdot U_k$$

$$\hat{P}_{k|k-1} = C \cdot P_{k-1|k-1} \cdot C^T + Q$$

$$inn_k = Y_k - \hat{Y}_k$$

$$S = A \cdot \hat{P}_{k|k-1} \cdot A^T + R$$

$$K = \hat{P}_{k|k-1} \cdot A^T \cdot S^{-1}$$

$$X_{k|k} = K \cdot inn_k + \hat{X}_{k|k-1}$$

$$P_{k|k} = (I - K \cdot A) \cdot \hat{P}_{k|k-1}$$

end

For (k=n; k>0; k--):

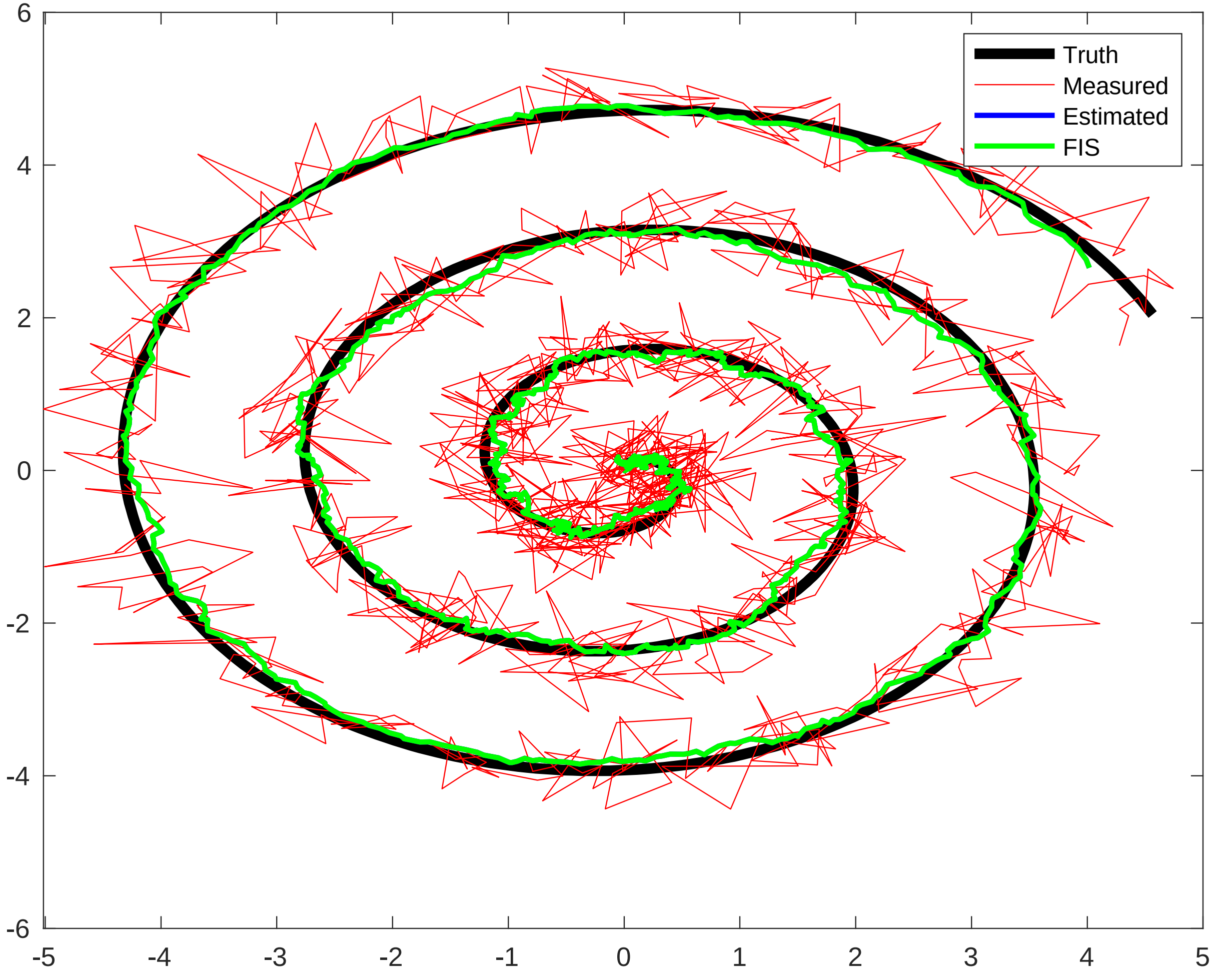
$$M_k = P_{k|k} \cdot A \cdot \hat{P}_{k+1|k}{}^{-1}$$

$$X_{k|n} = X_{k|k} + M_k \cdot (X_{k+1|n} - X_{k+1|k})$$

$$P_{k|n} = P_{k|k} + M_k \cdot (P_{k+1|n} - \hat{P}_{k+1|k})$$

end

Requires storing the $X_{k|k}$, $X_{k+1|k}$, $P_{k|k}$, and $\hat{P}_{k+1|k}$ from the forward pass



Truth
Measured
Estimated
FIS
6
4
2
0
-2
-4
-6
-5
-4
-3
-2
-1
0
1
2
3
4
5

Tuning R and Q

- R- Measurement noise
  - Uncertainty in each measurement
  - Low R → model will simply track the measurements (no filtering)
  - High Q → model will be independent of the data
  - Data fusion (use R to weight different measurements)

- Q- Process noise
  - Allowed changes in the state between updates
  - Low Q → state will be more static
  - High Q → allows state to vary wildly (less informative)
  - Ideally Q is the variance(diff(state))

Example

Observation Layer

$$\begin{bmatrix} Y_{lat}[n] \\ Y_{long}[n] \end{bmatrix} = \begin{bmatrix} X_{lat}[n] \\ X_{long}[n] \end{bmatrix} + \varepsilon$$

Position Layer

$$\begin{bmatrix} X_{lat}[n] \\ X_{long}[n] \end{bmatrix} = \begin{bmatrix} X_{lat}[n-1] \\ X_{long}[n-1] \end{bmatrix} + \mathrm{T} \cdot \begin{bmatrix} \cos(\theta[n]) \\ -\sin(\theta[n]) \end{bmatrix} \cdot V_{elocity}[n] + p$$

Velocity Layer

$$\begin{bmatrix} V_{elocity}[n] \\ \theta[n] \end{bmatrix} = \begin{bmatrix} V_{elocity}\ [n-1] \\ \theta[n-1] \end{bmatrix} + \mathrm{T} \cdot \begin{bmatrix} A_{cc}[n] \\ W_{acc}[n] \end{bmatrix} + v$$

Acceleration Layer

$$\begin{bmatrix} A_{cc}[n] \\ W_{acc}[n] \end{bmatrix} = \begin{bmatrix} A_{cc}[n-1] \\ W_{acc}[n-1] \end{bmatrix} + \omega$$

Example

$$Y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} X_{lat} \\ X_{long} \\ X_{velocity} \\ \theta_{velocity} \\ X_{accel} \\ \theta_{accel} \end{bmatrix} + \varepsilon$$

$$\begin{bmatrix} X_{lat} \\ X_{long} \\ X_{velocity} \\ \theta_{velocity} \\ X_{accel} \\ \theta_{accel} \end{bmatrix} = B\left(\begin{bmatrix} X_{lat} \\ X_{long} \\ X_{velocity} \\ \theta_{velocity} \\ X_{accel} \\ \theta_{accel} \end{bmatrix}\right) + \begin{bmatrix} \upsilon_{lat} \\ \upsilon_{long} \\ \upsilon_{velocity} \\ \upsilon_{velocity} \\ \upsilon_{accel} \\ \upsilon_{accel} \end{bmatrix}$$

$$B(X) = \begin{bmatrix} X_{lat} + \mathrm{T} * \cos(\theta_{velocity}) * X_{velocity} \\ X_{long} - \mathrm{T} * \sin(\theta_{velocity}) * X_{velocity} \\ X_{velocity} + T * X_{accel} \\ \theta_{velocity} + T * \theta_{accel} \\ 1 \\ 1 \end{bmatrix}$$

Example

$$Y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} X_{lat} \\ X_{long} \\ X_{velocity} \\ \theta_{velocity} \\ X_{accel} \\ \theta_{accel} \end{bmatrix} + \varepsilon$$

$$\begin{bmatrix} dX_{lat}/dt \\ dX_{long}/dt \\ dX_{velocity}/dt \\ d\theta_{velocity}/dt \\ dX_{accel}/dt \\ d\theta_{accel}/dt \end{bmatrix} = \begin{bmatrix} 1 & 0 & T \cdot \cos(\theta_{velocity}) & -T \cdot X_{velocity} \cdot \sin(\theta_{velocity}) & 0 & 0 \\ 0 & 1 & -\mathrm{T} \cdot \sin(\theta_{velocity}) & T \cdot X_{velocity} \cdot \cos(\theta_{velocity}) & 0 & 0 \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} dX_{lat}/dt \\ dX_{long}/dt \\ dX_{velocity}/dt \\ d\theta_{velocity}/dt \\ dX_{accel}/dt \\ d\theta_{accel}/dt \end{bmatrix} + \begin{bmatrix} \upsilon_{lat} \\ \upsilon_{long} \\ \upsilon_{velocity} \\ \upsilon_{velocity} \\ \upsilon_{accel} \\ \upsilon_{accel} \end{bmatrix}$$

Advantages-

- Model is restricted by real-world physics. (e.g. instantaneous velocity changes are not possible)
- More control over the sources of noise/error

$$Y[n] = H(X[n], U[n]) + e \qquad X[n] = F(X[n-1], U[n]) + v$$

$$\hat{X}_{pred}[n] = \mathrm{F}(\,\mathrm{X[n\text{-}1]}, U[n])$$

$$\hat{Y}_{pred}\,[n] = H(\hat{X}_{pred}[n], U[n])$$

$$\hat{P}_{pred}\,[n] = F' \cdot P[n] \cdot F'^{T} + Q$$

Prediction

$$F' = \left.\frac{\partial F}{\partial X}\right|_{\hat{X}_{pred}[n]}$$

$$err[n] = Y_{obs}[n] - \hat{Y}_{pred}\,[n]$$

$$S[n] = H' \cdot \hat{P}_{pred}\,[n] \cdot H'^{T} + R$$

$$K_{gain} = \hat{P}_{pred}\,[n] \cdot H'^{T} \cdot S[n]^{-1}$$

Update

$$H' = \left.\frac{\partial H}{\partial X}\right|_{\hat{Y}_{pred}\,[n]}$$

$$X[n] = K_{gain} \cdot err[n] + \hat{X}_{pred}[n]$$

$$P[n] = \left(I - K_{gain} \cdot H'\right) \cdot \hat{P}_{pred}\,[n]$$

Correction