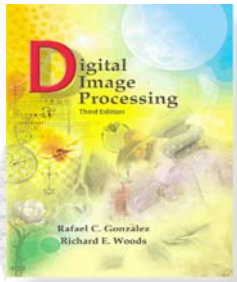
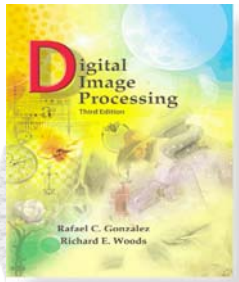


Lecture #27

- Mean Square Estimation*
- Kalman Filter**
- Kalman Tracking

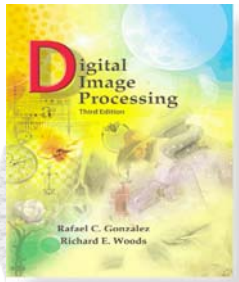
* See Section 8.3 Linear Mean-Square Estimation, Dwight F.Mix, Random Signal Processing, Prentice-Hall, 1995.

** See Section 8.6 Recursive Filtering, Dwight F.Mix, Random Signal Processing, Prentice-Hall, 1995.



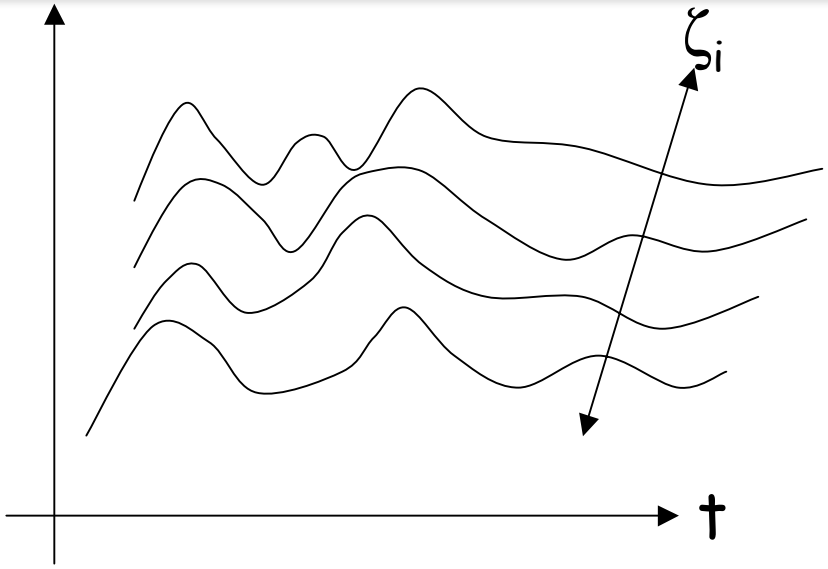
Stochastic processes & ensembles

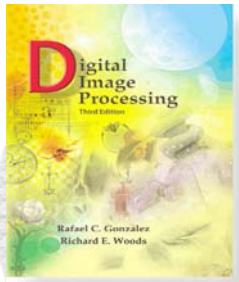
- A stochastic process produces an output waveform rather than just a number
- A specific output waveform is denoted by $X(t,\zeta_i)$



Stochastic processes & ensembles

- A collection of time functions $X(t,\zeta_i)$ is called an ensemble
- This illustrates an ensemble





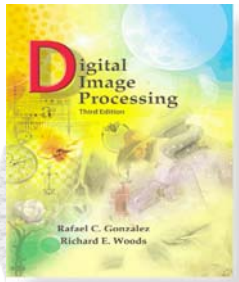
Mean Square Estimation

- Let $e = X - \hat{X}$ where e is the error between the random variable X and our estimate $\hat{X}$
- The mean squared error is:

$$E\left(e^2\right) = E\left[\left(X - \hat{X}\right)^2\right]$$

- The value of $\hat{X}$ which minimizes $E\left(e^2\right)$ is the minimum mean-square estimate of X

This is basically what we did in the Weiner filter



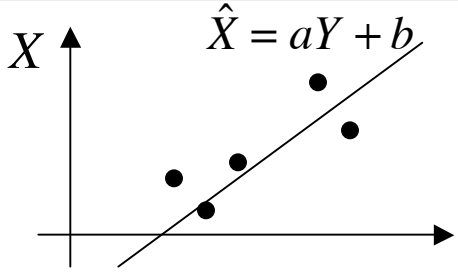
Estimating points on a line (1)

Y	4	1	0	1	4	9
X	-2	-1	0	1	2	3

Compute m_Y, σ_Y^2

Compute m_X, σ_X^2

Definition of a random variable X



Estimating a line through data points is an example of a minimum mean square estimator

- Estimate the value of X given Y by points on a straight line

$$\hat{X} = aY + b$$

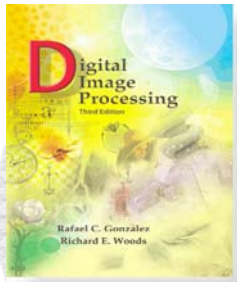
- Write the mean square error as

$$E\left(e^2\right) = E\left\{\left[X - \hat{X}\right]^2\right\} = E\left\{\left[X - (aY + b)\right]^2\right\}$$

- Set partial derivative of mean square error wrt b equal to zero to get b

$$\frac{\partial}{\partial b} E\left(e^2\right) = E\left\{2\left[X - aY - b\right](-1)\right\} = -2E(X) + 2aE(Y) + 2b = 0$$

$$b = E(X) - aE(Y) = m_X - am_Y$$



Estimating points on a line (2)

- Substituting our result for b into that for the error $E(e^2)$ we get

$$E\left(e^2\right) = E\left\{\left[X - aY - m_X + am_Y\right]^2\right\} = E\left\{\left[\left(X - m_X\right) - a\left(Y - m_Y\right)\right]^2\right\}$$

$$E\left(e^2\right) = E\left\{\left(X - m_X\right)^2 - 2a\left(X - m_X\right)\left(Y - m_Y\right) + a^2\left(Y - m_Y\right)^2\right\} = \sigma_X^2 - 2a\mu_{11} + a^2\sigma_Y^2$$

- Take the derivative wrt a and set equal to zero to get $a = \dfrac{\mu_{11}}{\sigma_Y^2}$
- We can calculate the means and variances of the data and, after substitution, get a and b as

$$a = \frac{\mu_{11}}{\sigma_Y^2} = 0.319 \qquad b = m_X - am_Y = 0.5 - \left(0.319\right)\left(3.17\right) = -0.5$$

where $\mu_{11} = E\left[\left(x - m_x\right)\left(y - m_y\right)\right]$

This gives $\hat{X} = 0.319Y - 0.5$

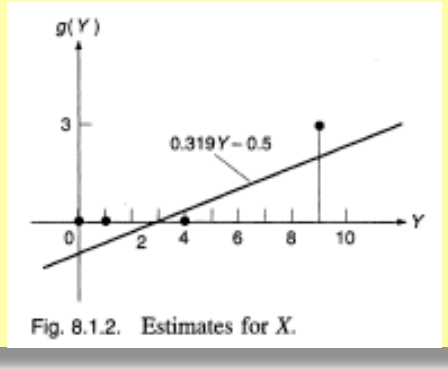
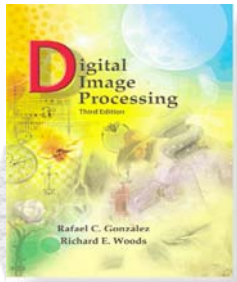


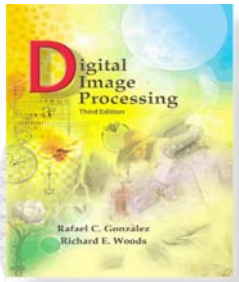
Fig. 8.1.2. Estimates for *X*.



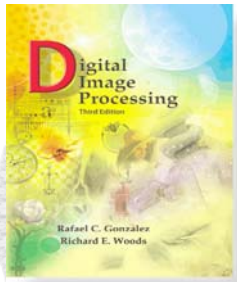
Continuous Waveform Measures

- The inner product $\langle v_1(t) \mid v_2(t) \rangle = \int_{-\infty}^{\infty} v_1(t) v_2(t) dt$
- The norm or length $\|v(t)\| = \sqrt{\int_{-\infty}^{\infty} v^2(t) dt}$
- Distance metric $d(v_1, v_2) = \sqrt{\int_{-\infty}^{\infty} [v_1(t) - v_2(t)]^2 dt}$

We want to treat waveforms like vectors and matrices



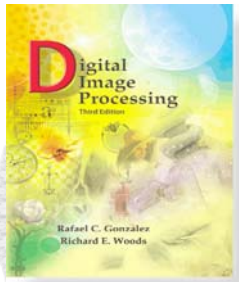
Discrete Waveform Calculations



Random Variable Measures

- The inner product $\langle X \mid Y \rangle = E(XY)$
- The norm or length $E(X) = \sqrt{E(X^2)}$
- Distance metric $d(X,Y) = \|X - Y\| = \sqrt{E\left((X-Y)^2\right)}$
- Orthogonality requires $\langle X \mid Y \rangle = E(XY) = 0$

Expand idea of functions as vectors to random variables. Note the use of E( ) throughout (see slide 46 of Lecture 25).

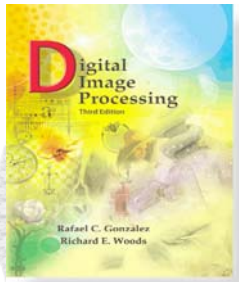


Random Variable Measures

- Now expand to discrete functions

$$E\left[g(x)\right] = \int_{-\infty}^{+\infty} g(x)\,p(x)\,dx$$ continuous

$$E\left[g(x)\right] = \sum_{i=1}^{N} g(x_i)\,P(x_i)$$ discrete



Linear estimator

We want to make an estimate of d which is a linear function of p+1 previously known, NOISY inputs

$$\hat{d} = h_0 x(n) + h_1 x(n-1) + h_2 x(n-2) + h_3 x(n-3) + \ldots + h_p x(n-p)$$

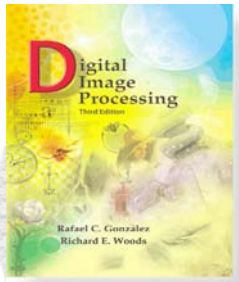
where x(i) is the data, the h_i's are constants, and $\hat{d}$ is the estimate of the output d

In general $x(n) = s(n) + w(n)$ where s is the actual signal and w is white noise

Extrapolation: $\hat{d}(n) = s(n+k)$ estimate a future value

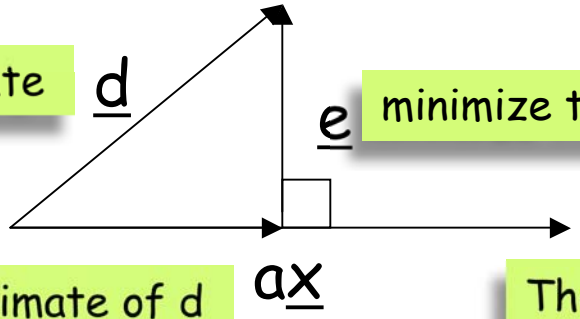
Interpolation: $\hat{d}(n) = s(n-k)$ estimate a previous value

Smoothing: $\hat{d}(n) = s(n)$ estimate the current value



Orthogonality Principle

This is what we want to estimate



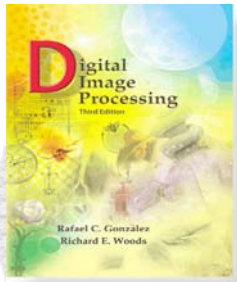
minimize the error by making $\underline{e} \perp \underline{x}$

This is the estimate of d from the measurements

These are the measurements

- The error $\underline{e}$ is the difference between the estimate $a\underline{x}$ and the actual $\underline{d}$
- We want to choose a so as to minimize $\underline{e}=\underline{d}-a\underline{x}$

For Kalman (and other) filters we minimize $\underline{e}$ by making $\underline{e} \perp \underline{x}$ where $\hat{d}=\underline{a} \cdot \underline{x}$, a linear estimator



Single Observation Estimation

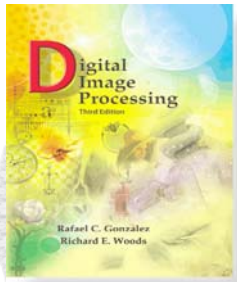
- (See Slides 61 and 62 of Lecture 25)
- The cross-correlation of x and y (the moment η_{11}) is given by

$$R_{xy} = \eta_{11} = E[xy] = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} xyp(x,y)\,dxdy$$

- and

$$E[d(n)x(n)] = E[s(n)x(n)] = R_{sx}(0)$$

This is always the difference between the times of the two waveforms— typically the second, x, minus the first, s.



Single Observation Estimation

- Given one observation x(n) we want to estimate s(n)

$$x(n) = s(n) + w(n) \qquad d(n) = s(n)$$

- Require the error $e(n) = d(n) - \hat{d}(n)$ to be orthogonal to the data x(n)

$$E\{e(n)x(n)\} = E\{(d(n) - \hat{d}(n))x(n)\} = 0$$

- Using the estimate $\hat{d}(n) = h_0 x(n)$ gives

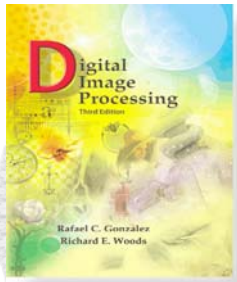
$$E\{(d(n) - h_0 x(n))x(n)\} = E\{d(n)x(n)\} - h_0 E\{x(n)x(n)\} = 0$$

- This can be re-arranged to give

$$E\{d(n)x(n)\} - h_0 E\{x(n)x(n)\} = R_{SX}(0) - h_0 R_{XX}(0) = 0$$

- Which says the optimum estimator occurs when

$$h_0 = \frac{R_{SX}(0)}{R_{XX}(0)}$$



Multiple Observations

Current sample and previous sample

- Given two observation x(n) and x(n-1) we want to estimate s(n)

$$x(n) = s(n) + w(n) \qquad d(n) = s(n)$$

- Require the error $e(n) = d(n) - \hat{d}(n)$ to be orthogonal to the data x(n) and x(n-1)

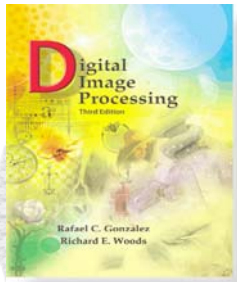
$$E\{e(n)x(n)\} = E\{(d(n) - \hat{d}(n))x(n)\} = 0$$

$$E\{e(n)x(n-1)\} = E\{(d(n) - \hat{d}(n))x(n-1)\} = 0$$

- Using the estimate $\hat{d}(n) = h_0 x(n) + h_1 x(n-1)$ now gives two equations since we require perpendicularity at both data points

$$E\{(d(n) - h_0 x(n) - h_1 x(n-1))x(n)\} = E\{d(n)x(n)\} - h_0 E\{x(n)x(n)\} - h_1 E\{x(n-1)x(n)\} = 0$$

$$E\{(d(n) - h_0 x(n) - h_1 x(n-1))x(n-1)\} = E\{d(n)x(n-1)\} - h_0 E\{x(n)x(n-1)\} - h_1 E\{x(n-1)x(n-1)\} = 0$$



Multiple Observations

- Rewriting these equations in terms of autocorrelation functions

$$E\{d(n)x(n)\} - h_0 E\{x(n)x(n)\} - h_1 E\{x(n-1)x(n)\} = R_{DX}(0) - h_0 R_{XX}(0) - h_0 R_{XX}(-1) = 0$$

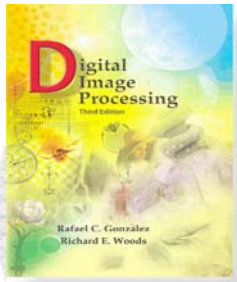
$$E\{d(n)x(n-1)\} - h_0 E\{x(n)x(n-1)\} - h_1 E\{x(n-1)x(n-1)\} = R_{DX}(1) - h_0 R_{XX}(1) - h_1 R_{XX}(0) = 0$$

- And putting them in matrix form gives

$$\begin{bmatrix} R_{XX}(0) & R_{XX}(-1) \\ R_{XX}(1) & R_{XX}(0) \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \end{bmatrix} = \begin{bmatrix} R_{DX}(0) \\ R_{DX}(1) \end{bmatrix} = \begin{bmatrix} R_{SX}(0) \\ R_{SX}(1) \end{bmatrix}$$

since s(x)=d(x)

- Which can be solved for h_0 and h_1.



Single Observation Example

Might be known analytically or a general model might be known

- Find the optimum h_0 and minimum mean-square error in estimating s(n) if the data is x(n)=s(n)+w(n). The noise w(n) is white Gaussian noise with zero mean and unit variance. The signal, which is zero mean and independent of the noise, has an autocorrelation function given by $R_{SS}(n)=0.9^{|n|}$
- The solution requires that we compute both $R_{XX}(0)$ and $R_{SX}(0)$.
- Computing $R_{XX}(0)$

$$R_{XX}(0) = E\{x(n)x(n)\} = E\{(s(n)+w(n))(s(n)+w(n))\}$$

$$R_{XX}(0) = E\{s(n)s(n)\} + E\{s(n)w(n)\} + E\{w(n)s(n)\} + E\{w(n)w(n)\}$$

$$R_{XX}(0) = R_{SS}(0) + R_{SW}(0) + R_{WS}(0) + R_{WW}(0)$$

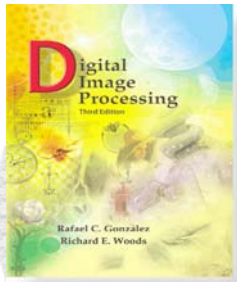
- Both cross-correlations are zero since the signal is independent of the noise and for white noise $R_{WW}(n)=\delta(n)$ giving:

Cross-correlations

$$R_{XX}(0) = R_{SS}(0) + R_{SW}(0) + R_{WS}(0) + R_{WW}(0) = 0.9^0 + 0 + 0 + \delta(0) = 1 + 1 = 2$$

Autocorrelation of signal

Autocorrelation of noise is δ(0)



Single Observation Example

- Computing $R_{SX}(0)$.

$$R_{SX}(0) = E\{s(n)x(n)\} = E\{s(n)(s(n)+w(n))\}$$
$$R_{SX}(0) = E\{s(n)s(n)\} + E\{s(n)w(n)\}$$
$$R_{SX}(0) = R_{SS}(0) + R_{SW}(0) = 0.9^0 + 0 = 1$$

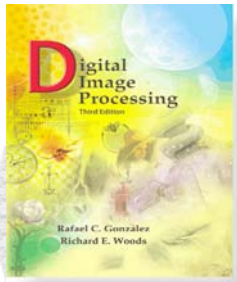
- We can evaluate the optimum estimator coefficient as

$$h_0 = \frac{R_{SX}(0)}{R_{XX}(0)} = \frac{1}{2}$$

- The mean squared error is given by

s(n)-$\hat{s}$(n) is simply the difference between the noise-free signal and the estimate from the noisy values

$$E(e^2) = E\left((s(n)-\hat{s}(n))(s(n)-\hat{s}(n))\right) = E\left((s(n)-h_0x(n))(s(n)-h_0x(n))\right)$$
$$E(e^2) = E(s(n)s(n)) - h_0E(s(n)x(n)) - h_0E(x(n)s(n)) + h_0^2E(x(n)x(n))$$
$$E(e^2) = R_{SS}(0) - h_0R_{SX}(0) - h_0R_{XS}(0) + h_0^2R_{XX}(0) = 0.9^0 - \left(\frac{1}{2}\right)1 - \left(\frac{1}{2}\right)1 + \left(\frac{1}{2}\right)^2 2 = \frac{1}{2}$$



Two Observation Example

- Expand the previous example to two observations, i.e., find the optimum h_0 and h_1 in estimating s(n) if the data is x(n)=s(n)+w(n). The noise w(n) is white Gaussian noise with zero mean and unit variance. The signal, which is also zero mean and is independent of the noise, has an autocorrelation function given by $R_{SS}(n)=0.9^{|n|}$
- The solution requires that we evaluate the matrix

$$\begin{bmatrix} R_{XX}(0) & R_{XX}(-1) \\ R_{XX}(1) & R_{XX}(0) \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \end{bmatrix} = \begin{bmatrix} R_{SX}(0) \\ R_{SX}(1) \end{bmatrix}$$

The big difference from the single observation example is that we need two equations and matrices.

- The new quantities to be evaluated are $R_{XX}(1)$, $R_{XX}(-1)$, and $R_{SX}(1)$.

$$R_{XX}(1) = R_{SS}(1) + R_{SW}(1) + R_{WS}(1) + R_{WW}(1) = 0.9^1 + 0 + 0 + \delta(1) = 0.9$$

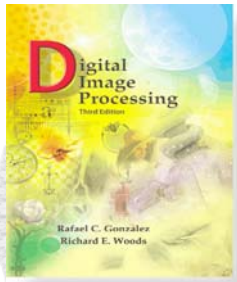
$$R_{XX}(-1) = R_{SS}(-1) + R_{SW}(-1) + R_{WS}(-1) + R_{WW}(-1) = 0.9^{|-1|} + 0 + 0 + \delta(1) = 0.9$$

$$R_{SX}(1) = E\{s(n)x(n+1)\} = E\{s(n)(s(n+1) + w(n+1))\}$$

$$R_{SX}(1) = E\{s(n)s(n+1)\} + E\{s(n)w(n+1)\}$$

$$R_{SX}(1) = R_{SS}(1) + R_{SW}(1) = 0.9^1 + 0 = 0.9$$

0



Two Observation Example

- Evaluating the matrices gives

$$\begin{bmatrix} 2 & 0.9 \\ 0.9 & 2 \end{bmatrix}\begin{bmatrix} h_0 \\ h_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.9 \end{bmatrix}$$

- Which can be solved to give h_0=0.3730 and h_1=0.2821. The mean-square error is calculated as

Remember s(n)-$\hat{s}$(n) is the difference between the noise-free signal and the estimate from the noisy values

$$E\left(e^2\right) = E\left(\left(s(n)-\hat{s}(n)\right)\left(s(n)-\hat{s}(n)\right)\right) = E\left(\left(s(n)-h_0x(n)-h_1x(n-1)\right)\left(s(n)-h_0x(n)-h_1x(n-1)\right)\right)$$

$$E\left(e^2\right) = E\left(s(n)s(n)\right)-h_0E\left(s(n)x(n)\right)-h_1E\left(s(n)x(n-1)\right)-h_0E\left(x(n)s(n)\right)+h_0^2E\left(x(n)x(n)\right)$$

$$+h_0h_1E\left(x(n)x(n-1)\right)-h_1E\left(x(n-1)s(n)\right)+h_0h_1E\left(x(n-1)x(n)\right)+h_1^2E\left(x(n-1)x(n-1)\right)$$

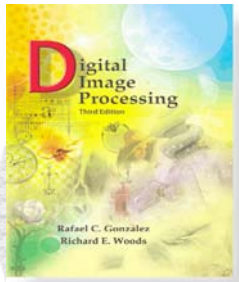
$$E\left(e^2\right) = R_{SS}(0)-h_0R_{SX}(0)-h_1R_{SX}(1)-h_0R_{XS}(0)+h_0^2R_{XX}(0)+h_0h_1R_{XX}(1)$$

$$-h_1R_{XS}(-1)+h_0h_1R_{XX}(-1)+h_1^2R_{XX}(0)$$

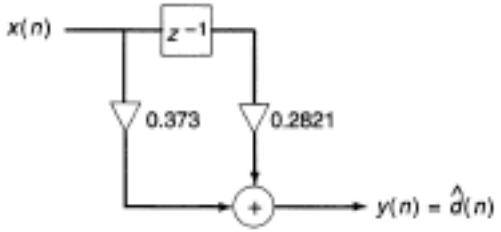
$$E\left(e^2\right) = 0.9^0-(0.373)(1)-(0.2821)(0.9)-(0.373)(1)+(0.373)^2(2)+(0.373)(0.2821)(0.9)$$

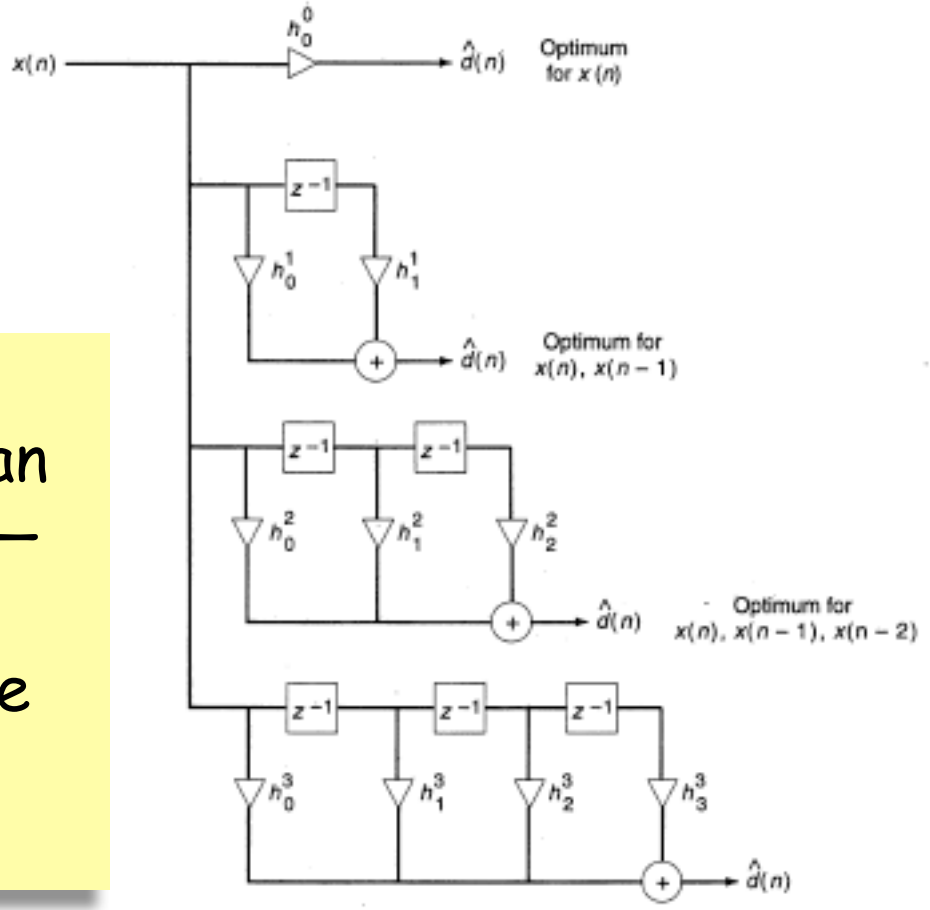
$$-(0.2821)(0.9)+(0.373)(0.2821)(0.9)+(0.2821)^2(2) = 0.373$$

$$R_{XS}(-1) = R_{SS}(-1)+R_{WS}(-1) = 0.9^{|-1|}+\delta(-1) = 0.9$$

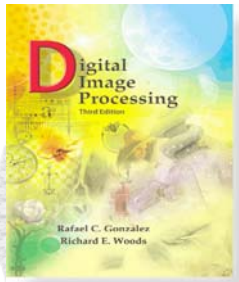


Estimator Filter Architecture



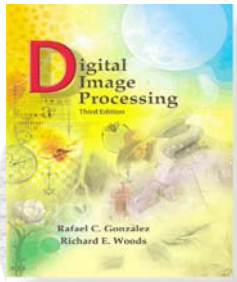


The two observation estimator example can be drawn as a filter — which can be generalized to include many more observations.



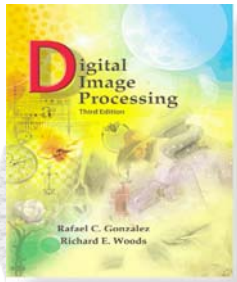
Kalman Filter

- What is the optimum estimator filter for n samples of a signal which is evolving over time?
- Kalman (1960) proposed a signal model which can be used to recursively estimate a signal evolving over time.



Optimum Filtering

- Kalman filters are often used to provide accurate estimates of position and velocity
- A Kalman filter is an efficient recursive filter which estimates the state of a dynamical system from a series of incomplete and noisy measurements
- Estimates can be
  - past time (interpolation or smoothing)
  - present time (filtering)
  - future time (prediction)

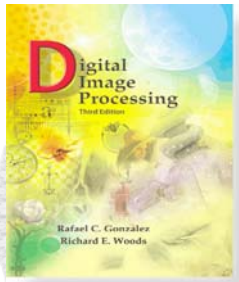


Design a Kalman Filter for a simple system

Simple model of the system that generated the data can be defined in multiple ways:

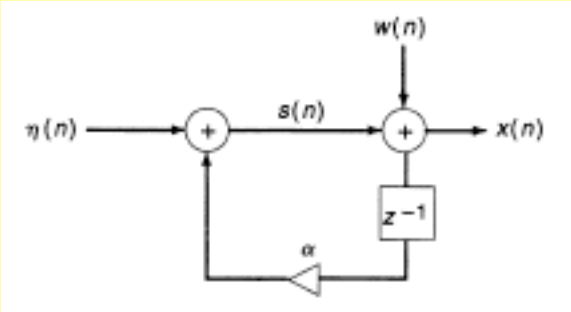
- Impulse function — $h(n) = \alpha^n u(n)$
- Transfer function — $H(z) = \dfrac{1}{1 - \alpha z^{-1}}$
- Difference equation — $s(n) = \alpha s(n-1) + \eta(n)$

This is probably the most commonly used model.



Kalman signal model

The Kalman signal model for this system is

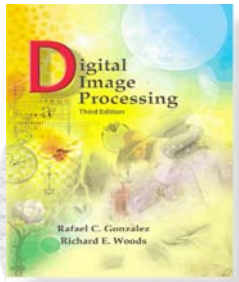


$\eta(n)$ is the white noise which drives the system

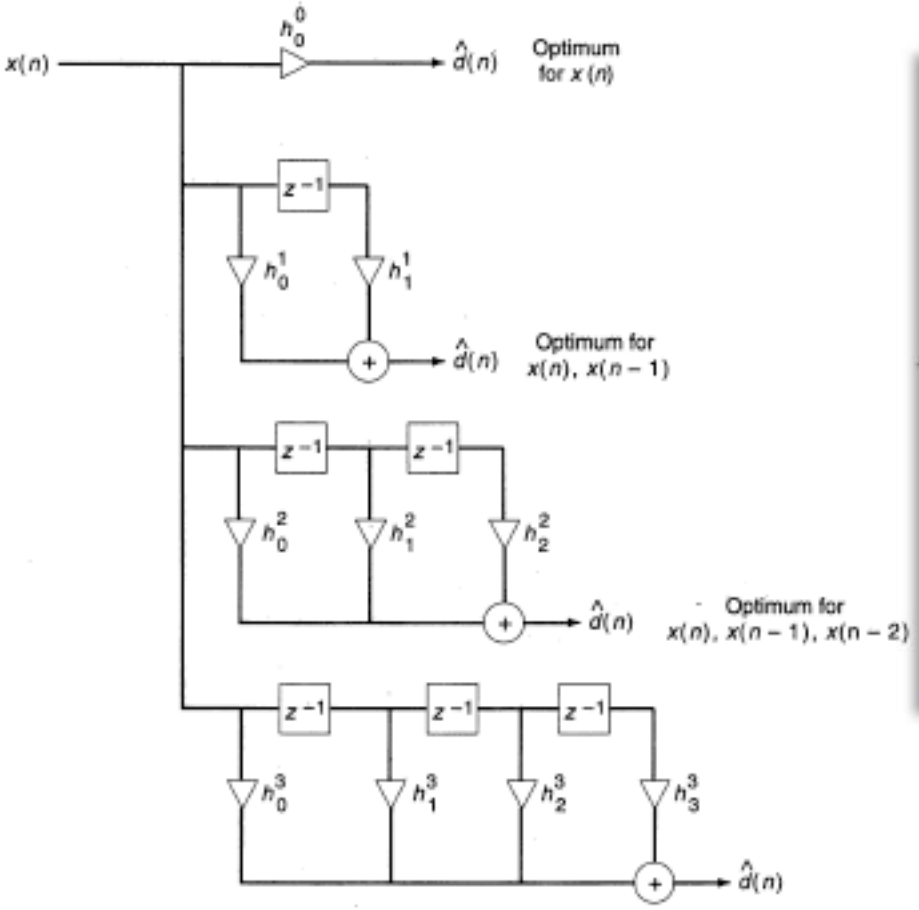
s(n) is the output signal

w(n) is the white noise in the observations and is independent of $\eta(n)$

x(n) is the actual observed output, i.e., s+n



Recursive Estimation?



In our previous architecture we needed to compute a new coefficient and add a delay (processing block)

Instead, can we recursively do a mean square estimate of the signal using the previous estimate and the new signal observation? If so, it would have to behave like

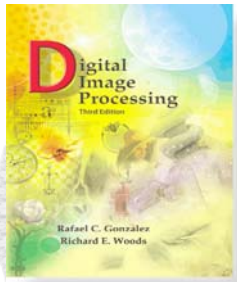
$$\hat{d}(n) = A_n \hat{d}(n-1) + K_n x(n)$$

new estimate

old estimate

new signal observation with a varying gain K_n

Can we use a fixed length (size) to recursively update the h_i^n?



The Kalman Filter

This is insight from Kalman

- Assume that we can write $A_n = (1 - K_n)\alpha$
- Then the optimum estimator can be written

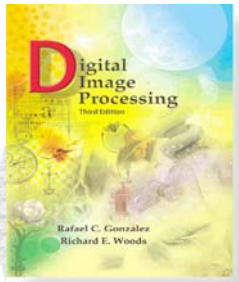
$$\hat{d}(n) = A_n \hat{d}(n-1) + K_n x(n) = (1 - K_n)\alpha \hat{d}(n-1) + K_n x(n)$$

- The normal form for this is

$$\hat{d}(n) = \alpha \hat{d}(n-1) + K_n \left[x(n) - \alpha \hat{d}(n-1) \right]$$

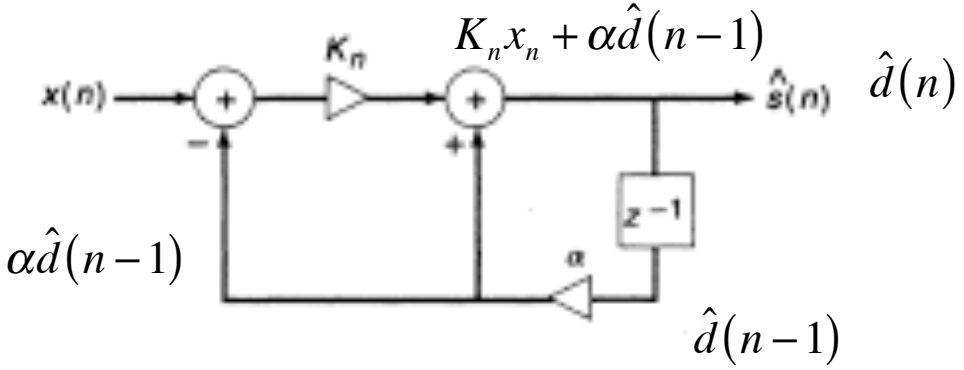
The purpose here is to get a specific architecture for implementation

- Where the first term is called the forward prediction term and the second is called the residual or correction term

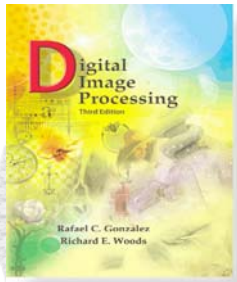


The Kalman Filter

- For the specified system, the Kalman filter uses a time varying gain K_n as shown below to set the error orthogonal to the signal



The key idea is that everything but K_n is constant. K_n is adjusted over time to keep e(n) orthogonal to x(n).



Basic Kalman theory

- The mean square error value is

$$\varepsilon(n) = E\left[e^2(n)\right] = E\left\{\left[d(n) - \hat{d}(n)\right]^2\right\} = E\left\{\left[d(n) - A_n\hat{d}(n-1) + K_n x(n)\right]^2\right\}$$

- Since we are using a mean-square estimator we want to set the error orthogonal to the data,i.e.,

$$\varepsilon(n) = E\left[e(n)d(n)\right]$$

- Solving this (without proof) requires

$$K_n = \frac{\varepsilon(n)}{E\left[\left(w(n) - m_w\right)^2\right]} = \frac{\varepsilon(n)}{\sigma_W^2}$$

This is the critical Kalman formula for the gain K_n.

- Where

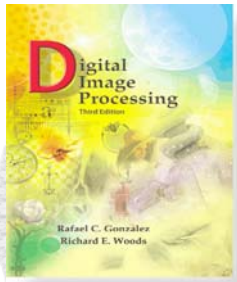
$$\varepsilon(n) = \left[\frac{\sigma_\eta^2 + \alpha^2\varepsilon(n-1)}{\sigma_\eta^2 + \sigma_W^2 + \alpha^2\varepsilon(n-1)}\right]\sigma_W^2$$

ε(n) is the error which changes over time .

- With initial value

$$\varepsilon(0) = \frac{\sigma_S^2\sigma_W^2}{\sigma_S^2 + \sigma_W^2}$$

σ_w^2 is the variance of the noise which does not changeover time.



Kalman Filter algorithm

The signal has an exponential autocorrelation function. The parameters α and σ_η^2 must be known. The additive noise w(n) is white with known variance σ_W^2. Then

$$K_n = \varepsilon(n) \Big/ \sigma_W^2 \qquad \varepsilon(0) = \frac{\sigma_S^2 \sigma_W^2}{\sigma_S^2 + \sigma_W^2}$$

Step 1. Set n=0 and calculate the initial mean square error

Step 2. Calculate the Kalman gain

Step 3. Input the data x(n) and calculate the new estimate.

$$\hat{s}(n) = \alpha \hat{s}(n-1) + K_n \left[x(n) - \alpha \hat{s}(n-1) \right]$$

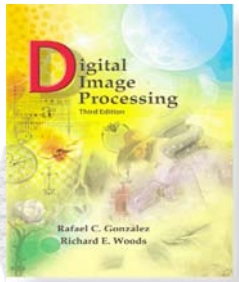
- For n=0 assume $\hat{s}(0) = 0$ so that $\hat{s}(0) = K_n x(0)$

Step 4. Let n=n+1

Step 5. Update the error $\varepsilon(n) = \left[\frac{\sigma_\eta^2 + \alpha^2 \varepsilon(n-1)}{\sigma_\eta^2 + \sigma_W^2 + \alpha^2 \varepsilon(n-1)} \right] \sigma_W^2$

- where $\sigma_\eta^2 = (1 - \alpha^2) \sigma_S^2$

Step 6. Go to Step 2.



Example of Kalman Filtering

Consider a particle moving in the plane at constant velocity subject to random perturbations in its trajectory. The new position (x_1, x_2) is the old position plus the velocity $(\Delta x_1, \Delta x_2)$ X1 plus noise w

$$\begin{bmatrix} x_1(t) \\ x_2(t) \\ \Delta x_1(t) \\ \Delta x_2(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ \Delta x_1(t) \\ \Delta x_2(t) \end{bmatrix} + \begin{bmatrix} w_{x1} \\ w_{x2} \\ w_{\Delta x1} \\ w_{\Delta x2} \end{bmatrix}$$

We assume we can only measure the position of the particle

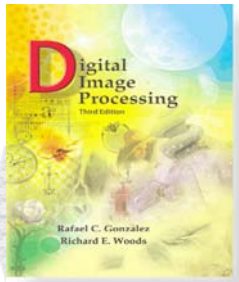
This will be called the A matrix

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ \Delta x_1(t) \\ \Delta x_2(t) \end{bmatrix} + \begin{bmatrix} v_{x1} \\ v_{x2} \end{bmatrix}$$

new position x,y

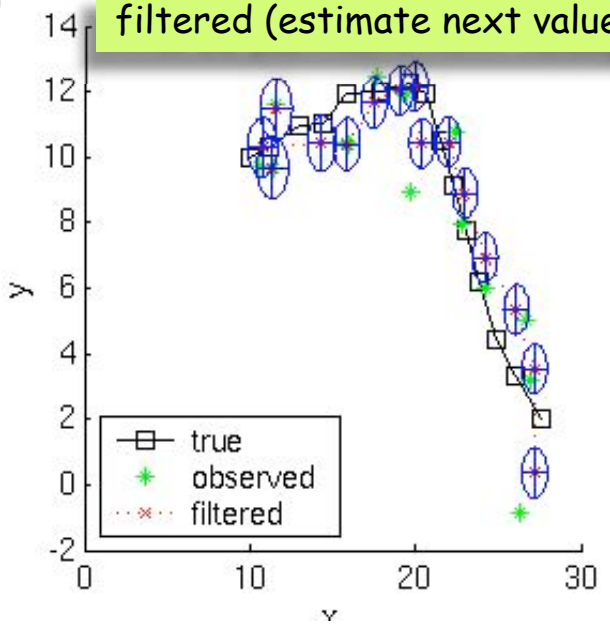
velocities v_x, v_y

This will be called the H matrix

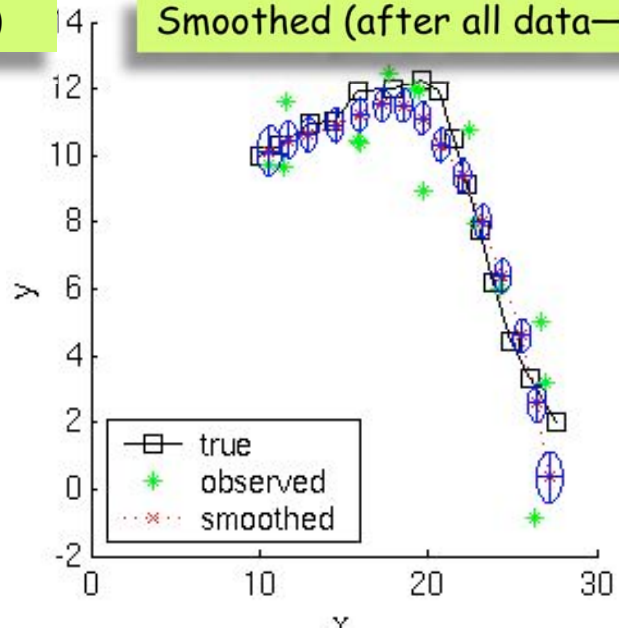


Example of Kalman Filtering

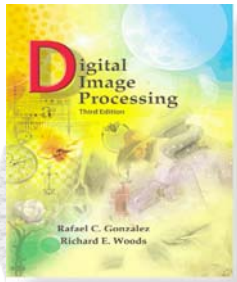
filtered (estimate next value)



Smoothed (after all data—curve fit)



Suppose we start out at position (10,10) moving to the right with velocity (1,0). We sampled a random trajectory of length 15. The figures show the filtered and smoothed trajectories. The mean squared error of the filtered estimate is 4.9; for the smoothed estimate it is 3.2. Not only is the smoothed estimate better, but we know that it is better, as illustrated by the smaller uncertainty ellipses



Kalman Filter algorithm (matrix formulation)

Predict the next state s from the initial state

$$\hat{s}_1 = As_0$$

Update the covariance matrix P for the predicted state

$$\hat{P}_1 = AP_0A^T + Q$$

Q is the noise inherent to the variables

Look for the location of the new feature and measure it. This is m_1
Compute the Kalman gain using this m_1 and update the covariance matrix.

$$K_1 = \hat{P}_1 H^T \left(H\hat{P}_1 H^T + R \right)^{-1}$$

R is the measurement noise, i.e., error

$$m_1 = Hs_0$$

measurement

Initial state (previous measurement)

$$P_1 = \hat{P}_1 + K_1 H\hat{P}_1$$

Now compute the new prediction for the next state using the Kalman gain K_1

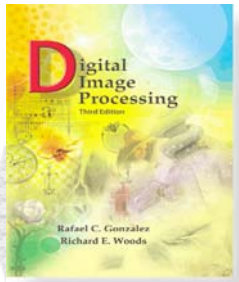
New state

$$\hat{s}_2 = \hat{s}_1 + K_1 \left(m_1 - H\hat{s}_1 \right)$$

And repeat the process.

measurement

Previous state

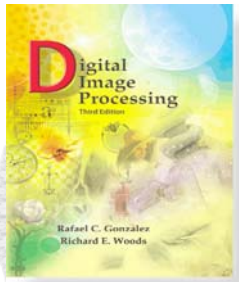


Kalman Filtering applied to Truck Tracking

Using the matrix formulation we just developed.

Image Processing and Interpretation Center, Steve Mills and Tony Pridmore, The University of Nottingham

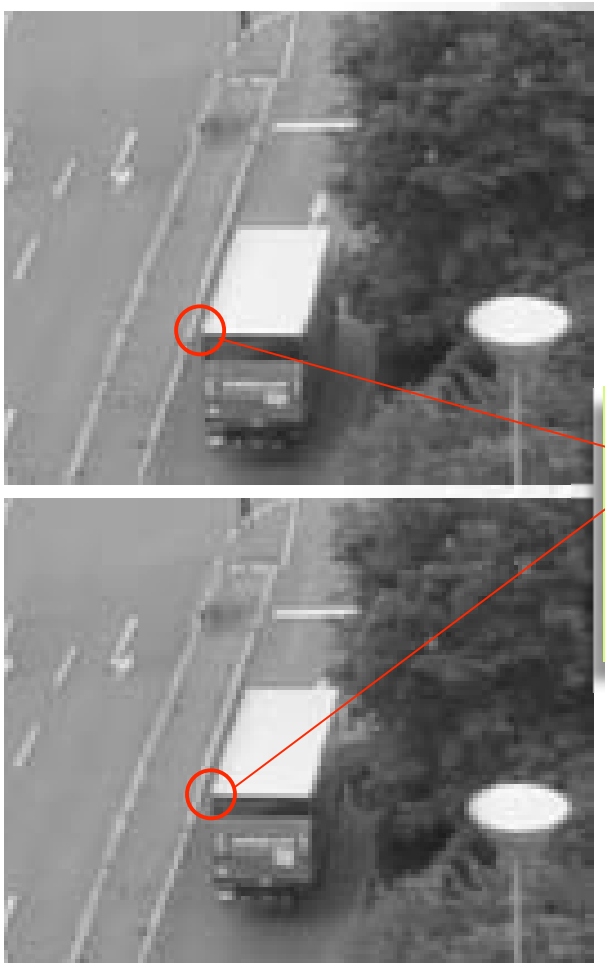
http://www.cs.nott.ac.uk/~tpp/G5BVIS/lectures.html



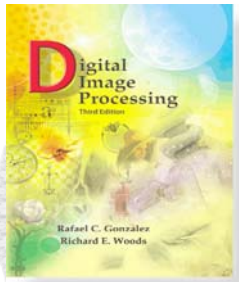
Application - Traffic Tracking

We want to track vehicles on a road

- Eg: The truck in the images to the right
- They are moving with a (fairly) constant velocity
- In each frame we can measure the position of a feature on the vehicle we want to track



Track lower left hand corner of truck



State Update Equation

- We assume the truck is moving with a constant velocity
- Our state is the truck position (x,y) and corresponding velocities (u,v)

$$s = \begin{bmatrix} x \\ y \\ u \\ v \end{bmatrix}$$

- At each time the velocity changes the position

$$x_t = x_{t-1} + u_{t-1} \times 1$$
$$y_t = y_{t-1} + v_{t-1} \times 1$$

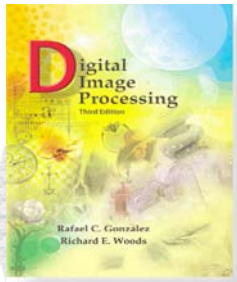
Time between steps

$$u_t = u_{t-1}$$
$$v_t = v_{t-1}$$

$$\begin{bmatrix} x_t \\ y_t \\ u_t \\ v_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ u_{t-1} \\ v_{t-1} \end{bmatrix}$$

$$s_t = As_{t-1}$$

State update equation



Measurement Equation

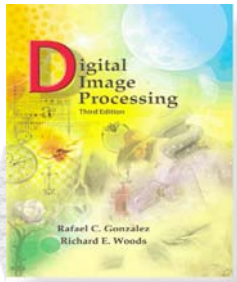
- At each time we can detect features in the image
- These make our measurements, m_t
- We can directly measure the position of the truck, but not the velocity

$$m_t = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ u_{t-1} \\ v_{t-1} \end{bmatrix}$$

$$m_t = Hs_t$$

Measurement equation



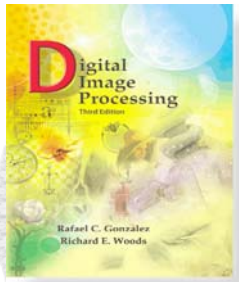
An Initial Estimate

- The initial estimate of the state
  - We give a rough value of x and y to say which feature we are tracking
  - We probably won't have any idea about u and v
  - So we will use

$$s_0 = \begin{bmatrix} 100 \\ 170 \\ 0 \\ 0 \end{bmatrix}$$

Initial measurement of corner

- We also need to give the (un) certainty
  - Our estimate of the position is good to within a few pixels
  - Our motion estimate is not good, but we expect the motion to be small
  - We represent this as a covariance matrix

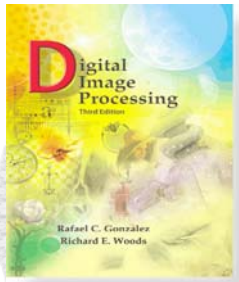


Covariance Matrices

- So what is a covariance matrix?
  - It gives the relationships between sets of variables
  - The variance of a variable, x, is $Var(x)=E\{(x-\bar{x})^2\}$
  - The covariance of two variables, x and y, is $Cov(x,y)=E\{(x-\bar{x})(y-\bar{y})\}$

$$\begin{array}{c} \\ x \\ y \\ u \\ v \end{array} \begin{array}{c} \begin{array}{cccc} x & y & u & v \end{array} \\ \begin{bmatrix} C_{xx} & C_{xy} & C_{xu} & C_{xv} \\ C_{yx} & C_{yy} & C_{yu} & C_{yv} \\ C_{ux} & C_{uy} & C_{uu} & C_{uv} \\ C_{vx} & C_{vy} & C_{vu} & C_{vv} \end{bmatrix} \end{array}$$

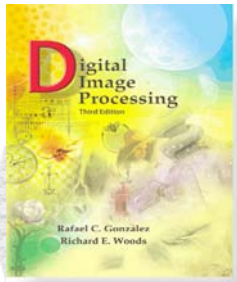
$$where \quad C_{xy} = E\left[(x-\bar{x})(y-\bar{y})\right], \; etc.$$



Covariance Matrices

$$\begin{array}{c} \\ x \\ y \\ u \\ v \end{array} \begin{array}{c} \begin{array}{cccc} x & y & u & v \end{array} \\ \begin{bmatrix} C_{xx} & C_{xy} & C_{xu} & C_{xv} \\ C_{yx} & C_{yy} & C_{yu} & C_{yv} \\ C_{ux} & C_{uy} & C_{uu} & C_{uv} \\ C_{vx} & C_{vy} & C_{vu} & C_{vv} \end{bmatrix} \end{array}$$

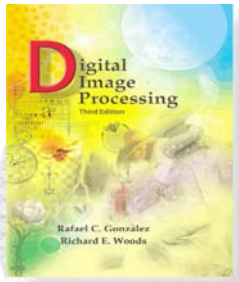
- Given a vector of variables $x=[x_1,x_2,\ldots,x_k]$
  - The covariance, *C*, is a *kxk* matrix
  - The i,j^{th} entry of *C* is $C_{ij}=\text{cov}(x,y)$
  - A diagonal entry, C_{ii}, gives the variance in the variable x_i
  - *C* is symmetric



Covariance in Noise

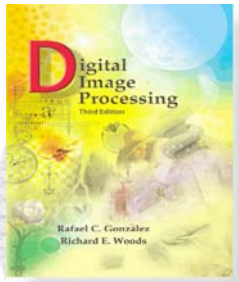
- The noise terms v (the measurement noise or error) and w (the process noise) need to be estimated.
  - They have zero mean, and covariance matrices R and Q respectively.
  - We need an estimate of these matrices. Q and R say how certain we are about our model equations.

- To estimate Q (the process noise)
  - Our initial estimate will be within a few pixels, say $\sigma=3$
  - The velocity is a bit less certain, but won't be large, say $\sigma=5$
  - There is no reason to think that the errors are related, so the covariance terms will be zero



Covariance in Noise

w	process noise (white)
v	measurement error
Q	covariance of process noise (4x4 since there are 4 state variables)
R	covariance of measurement error (2x2 since only two measurements)



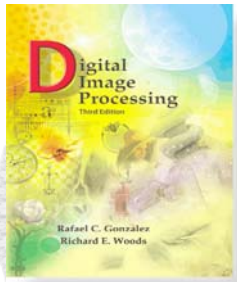
a priori Estimate Covariance Matrix

- The variances of x and y are $3^2=9$
- The variances of u and v are $5^2=25$
- Since we assume independence the off-diagonal entries are all 0

These are assumptions

$$P_0 = \begin{bmatrix} 9 & 0 & 0 & 0 \\ 0 & 9 & 0 & 0 \\ 0 & 0 & 25 & 0 \\ 0 & 0 & 0 & 25 \end{bmatrix}$$

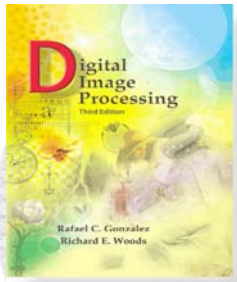
This represents the variances of the original state variables



Uncertainty in the model

- Our model equations have noise terms
  - v represents the fact that our state update model may not be accurate
  - w represents the fact that measurements will always be noisy
  - We need to estimate their covariances

- In general
  - Often the terms will be independent. If this is the case the off-diagonal entries will be zero
  - Choosing the diagonal entries (variances) is often more difficult



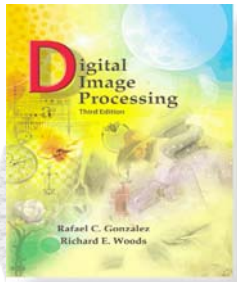
Process Noise Covariance Q

- The state update equation is not perfect
  - It assumes that the motion is constant but *u* and *v* might change over time
  - It assumes that all the motion is represented by *u* and *v* but other factors might affect *x* and *y*

- These errors will probably be small
  - The motion is slow and quite smooth
  - So the variance in these terms is probably a pixel or less, say $\sigma=1/2$

$$Q = \begin{bmatrix} 0.25 & 0 & 0 & 0 \\ 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0.25 & 0 \\ 0 & 0 & 0 & 0.25 \end{bmatrix}$$

Assume the same variance ($\sigma=1/2$ pixel) for all variables

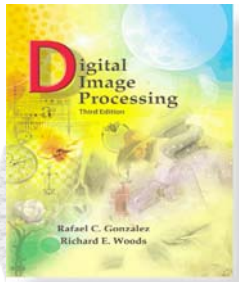


Measurement Error Covariance R

- The measurements we make will not be perfect
  - The features are located only to the nearest pixel
  - Because of image noise, aliasing, etc. they might be off by a pixel or so

- These errors are a bit easier to estimate
  - The feature is probably in the right place, or a pixel off
  - So the variance in these terms is probably $\sigma^2=1$

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



1. Predict the State

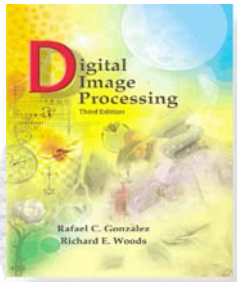
- We can now run the filter
  - First we make a prediction of the state at t=1 based on our initial estimate at t=0

Initial position

$$\bar{s}_1 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 100 \\ 170 \\ 0 \\ 0 \end{bmatrix}$$

$$\bar{s}_1 = \begin{bmatrix} 100 \\ 170 \\ 0 \\ 0 \end{bmatrix}$$

Prediction of state, i.e., the next point—this is where we will look for the next state



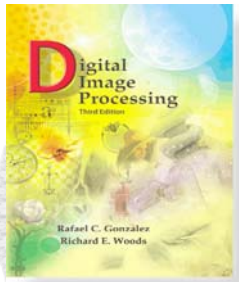
2. Update the *a priori* Prediction Covariance

$$P_1^- = AP_0A^T + Q$$

$$P_1^- = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 9 & 0 & 0 & 0 \\ 0 & 9 & 0 & 0 \\ 0 & 0 & 25 & 0 \\ 0 & 0 & 0 & 25 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0.25 & 0 & 0 & 0 \\ 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0.25 & 0 \\ 0 & 0 & 0 & 0.25 \end{bmatrix}$$

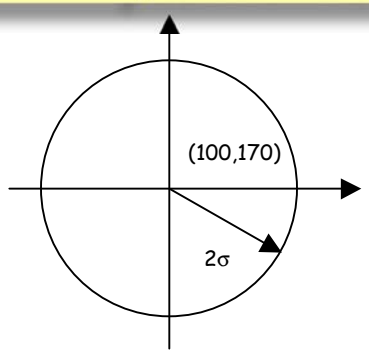
$$P_1^- = \begin{bmatrix} 34.25 & 0 & 25 & 0 \\ 0 & 34.25 & 0 & 25 \\ 25 & 0 & 35.25 & 0 \\ 0 & 25 & 0 & 25.25 \end{bmatrix}$$

Now update the estimate (prediction) covariance. *A priori* is before measurement. This tells us how far to look for the next measurement.



3a. Look for the Next Point

- The state prediction gives us a guide to where the feature will be
  - We expect it to be near (100,170)
  - The variance in the *x* position is 34.25
  - The variance in the *y* position is also 34.25

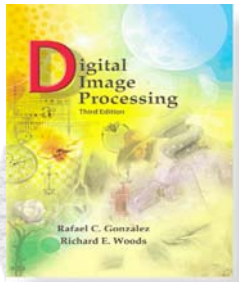


Look for next measurement of feature inside this circle. Circular since $\sigma_x^2=\sigma_y^2=34.25$

- We can use this to restrict our search for a feature
  - We are 95% certain that the feature lies in a circle of radius of 2σ of the prediction

$$\sigma = \sqrt{34.25} \approx 5.85$$

  - We look for a feature in this region



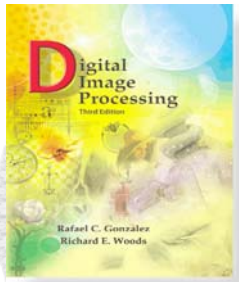
3b. Making the Actual Measurement

Find the corner using image processing

Within the search region

- We compute a value that tells us how likely each point is to be a feature (Harris interest operator)
- We find the point with the largest value within this region
- This is $m_1 = \begin{bmatrix} 103 \\ 163 \end{bmatrix}$

We look for a feature near our predicted value, and the covariances tell us how widely to search



4. Compute the Kalman Gain

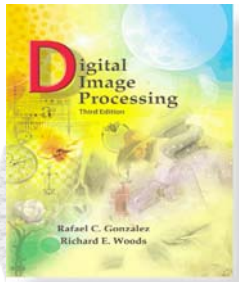
- We now combine the prediction and measurement
  - We compute the Kalman gain matrix
  - This takes into account the relative certainty of the two pieces of information

This begins the measurement update ("correction") phase

$$K_1 = P_1^- H^T \left(H P_1^- H^T + R \right)^{-1}$$

$$K_1 \approx \begin{bmatrix} 0.972 & 0 \\ 0 & 0.972 \\ 0.709 & 0 \\ 0 & 0.709 \end{bmatrix}$$

- The first components are close to 1, which will give more trust to the measurement



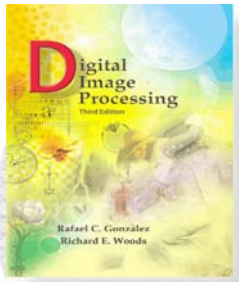
5. Update the *a posteriori* error Covariance

$$P_1 = P_1^- + K_1 H P_1^-$$

$$P_1 \approx \begin{bmatrix} 34.25 & 0 & 25 & 0 \\ 0 & 34.25 & 0 & 25 \\ 25 & 0 & 25.25 & 0 \\ 0 & 25 & 0 & 25.25 \end{bmatrix} + \begin{bmatrix} 0.972 & 0 \\ 0 & 0.972 \\ 0.709 & 0 \\ 0 & 0.709 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 34.25 & 0 & 25 & 0 \\ 0 & 34.25 & 0 & 25 \\ 25 & 0 & 25.25 & 0 \\ 0 & 25 & 0 & 25.25 \end{bmatrix}$$

$$P_1^- = \begin{bmatrix} 0.971 & 0 & 0.71 & 0 \\ 0 & 0.971 & 0 & 0.71 \\ 0.71 & 0 & 7.52 & 0 \\ 0 & 0.71 & 0 & 7.52 \end{bmatrix}$$

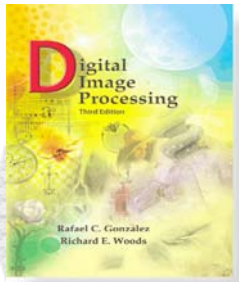
This is the updated covariance based upon the measurements just made.



6. Update estimate with measurement m_k

- The new (*a posteriori*) state estimate based upon measurement m_k is then

$$\hat{s}_k = \hat{s}_k^- + K_k\left(m_k - H\hat{s}_k^-\right)$$



Iteration

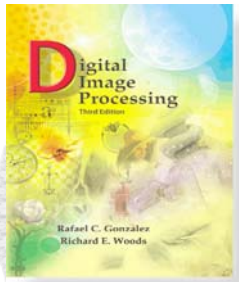
- We compute the next state

$$\hat{s}_k^- = \hat{s}_{k-1} + Bu_{k-1}$$

- And project the error covariance ahead

$$P_k^- = AP_{k-1}A^T + Q$$

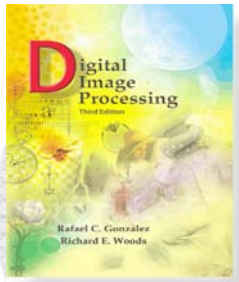
This tells us where to look next.



Iteration

- We repeat this cycle for each frame
- Over time the state predictions become more accurate
- The Kalman gain takes this into account and places more weight on the predictions

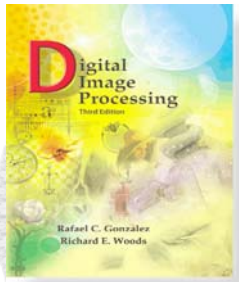
- To implement the Kalman filter
  - We need a lot of matrix subroutines
  - These are tiresome to code by hand, but there are several libraries available
  - Only need basic operations: +, -,x, transpose, and inverse



The Extended Kalman Filter

- The Kalman filter is limited by its assumptions
  - It assumes that all the noise/error terms are Gaussians with known (co)variance
  - It assumes that the model equations are linear

- Extended Kalman filters overcome the second assumption
  - They use a linear approximation to a non-linear function
  - They depend on the accuracy of this approximation
  - No proof, but they work well in practice

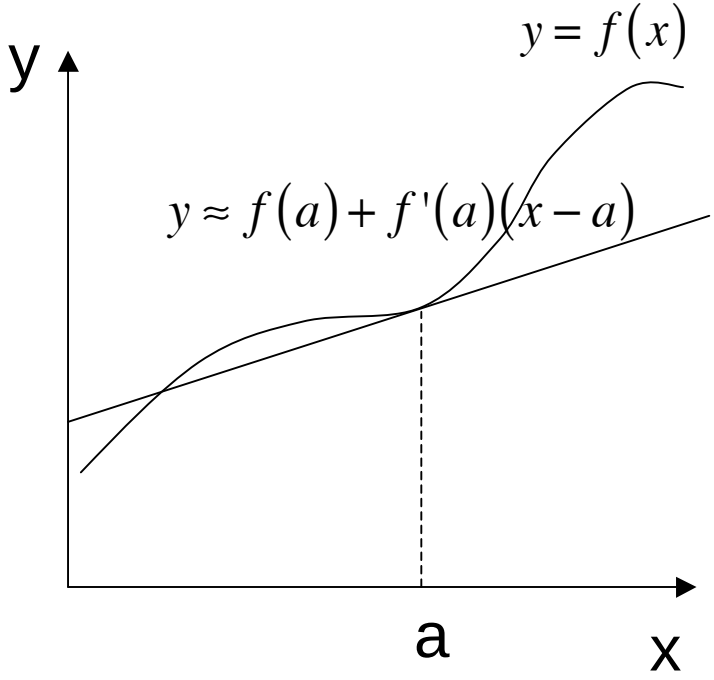


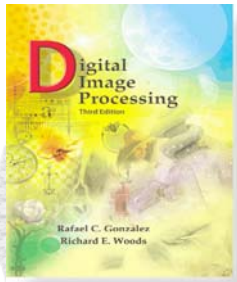
Linear Approximations

- If we have some function, y=f(x)
  - We can approximate this using

$$y \approx f(a) + f'(a)(x - a)$$

  - A is any value we choose
  - This approximation is best when $x \approx a$





Linear Approximations

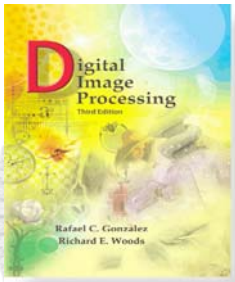
- If we have $z=f(x,y)$ we get

$$z \approx f(a,b) + f_x(a,b)(x-a) + f_y(a,b)(y-b)$$

  - fx and fy are the partial derivatives of f with respect to x and y
  - This approximates a 2D surface by a plane

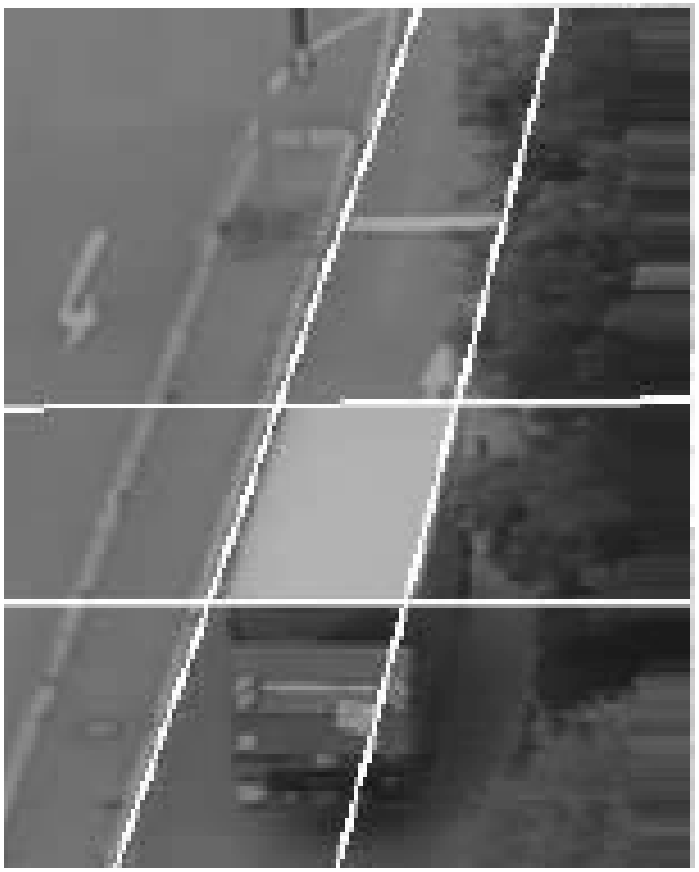
- More generally, given $y=f(x_1,x_2,\ldots,x_k)$ we have

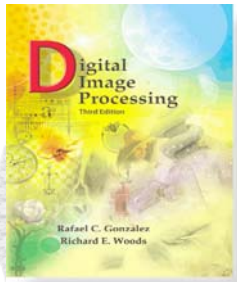
$$y \approx f(a_1,a_2,\ldots,a_k) + f_{x1}(a_1,a_2,\ldots,a_k)(x_1-a_1) + f_{x2}(a_1,a_2,\ldots,a_k)(x_2-a_2) + \ldots + f_{xk}(a_1,a_2,\ldots,a_k)(x_k-a_k)$$



Example EKF

- Lines are detected with the Hough transform
  - The relationship between the states at subsequent times is non-linear
  - An extended Kalman filter allows us to track groups of lines with a common motion model





For more information

- Tracking lines with the EKF

  Tracking in a Hough Space with the Extended Kalman Filter, Steven Mills, Tony Pridmore, and Mark Hills, Proceedings of the British Machine Vision Conference (BMVC2003), pages 173-182, 2003.

- A useful Java matrix library is JAMA

  http://math.nist.gov/javanumerics/jama

- One of the best introductory Kalman filter papers is Greg Welch and Gary Bishop, “An Introduction to the Kalman Filter,” TR 95-041,University of North Carolina at Chapel Hill.