Digital Image Processing
Lectures 23 & 24

M.R. Azimi, Professor

Department of Electrical and Computer Engineering
Colorado State University

Colorado State University

Restoration Filters

There are basically two classes of restoration filters.

1. **Deterministic-Based**
   These methods ignore effects of noise and statistics of the image, e.g., inverse filter and Least Squares (LS) filter.
2. **Stochastic-Based**
   Statistical information of the noise and image is used to generate the restoration filters, e.g., 2-D Wiener filter and 2-D Kalman filter.

Inverse Filter

(a) Direct Inverse Filter: Attempts to recover the original image from the observed blurred image using an inverse system, $h^I(m,n)$, corresponding to the blur PSF, $h(m,n)$.

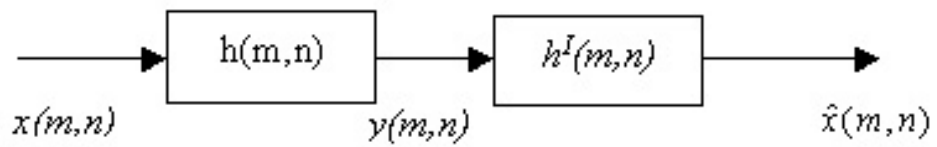


Figure 1: Inverse Filtering.

If we assume no noise case, we have

$$\begin{aligned} y(m,n) &= h(m,n) * *x(m,n) \\ Y(k,l) &= H(k,l)X(k,l) \end{aligned}$$

The inverse filter produces

$$\begin{aligned} \hat{x}(m,n) &= y(m,n) * *h^I(m,n) \\ \hat{X}(k,l) &= Y(k,l)H^I(k,l) \end{aligned}$$

Then,

$$\hat{x}(m,n) = x(m,n) * *h(m,n) * *h^I(m,n)$$

or

$$\hat{X}(k,l) = X(k,l)H(k,l)H^I(k,l)$$

Clearly, $\hat{x}(m,n) = x(m,n)$ or $\hat{X}(k,l) = X(k,l)$ iff $H^I(k,l) = \frac{1}{H(k,l)}$ or $h(m,n) * *h^I(m,n) = \delta(m,n)$.

Thus

$$\hat{X}(k,l) = Y(k,l)/H(k,l)$$

Now, if there is a slight noise (e.g., quantization noise) in the image,

$$Y(k,l) = H(k,l)X(k,l) + N(k,l)$$

The inverse filter gives

$$\hat{X}(k,l) = X(k,l) + \frac{N(k,l)}{H(k,l)}$$

At those frequencies where $H(k,l) \simeq 0$, $\frac{N(k,l)}{H(k,l)}$ becomes very large i.e. the noise is amplified.

(b) Pseudo Inverse Filter

To overcome the problems with the direct inverse filter, modify the transfer function of the inverse filter as

$$H^I(k,l) = \frac{H^*(k,l)}{|H(k,l)|^2 + \varepsilon}$$

where ε is a small positive quantity. For $\varepsilon = 0$, we have $H^I(k,l) = \frac{1}{H(k,l)}$. Alternatively, we can use

$$H^+(k,l) = \begin{cases} \frac{1}{H(k,l)} & |H(k,l)| \geq \varepsilon \\ 0 & |H(k,l)| < \varepsilon \end{cases}$$

While the first form of the pseudo inverse filter corresponds to a special case of Wiener filter (discussed next) it does not take into account the statistics of the noise and image.

Example 1: Derive the transfer function of the LS filter and show that it is the same as the inverse filter.

Solution: The LS solution minimizes the LS cost function,

$$J(\hat{x}) = \sum_m \sum_n |y(m,n) - \hat{y}(m,n)|^2$$

where $\hat{y}(m,n) = h(m,n) * *\hat{x}(m,n)$ is an estimate of the observed image based upon the LS estimate $\hat{x}(m,n)$. Using energy preservation,

$$J(\hat{x}) = J(\hat{X}) = \frac{1}{N^2} \sum_k \sum_l |Y(k,l) - \hat{Y}(k,l)|^2$$

where $\hat{Y}(k,l) = H(k,l)\hat{X}(k,l)$. Taking derivative of J wrt $\hat{X}(k,l)$ gives

$$\frac{\partial J}{\partial \hat{X}(k,l)} = 0 \Longrightarrow -H^*(k,l)(Y(k,l) - H(k,l)\hat{X}(k,l)) = 0$$

which gives the same inverse filter solution i.e.,

$$\hat{X}_{LS}(k,l) = Y(k,l)/H(k,l)$$

Quest ion: What would you change in the cost function to yield LS solution that is the same as that of the pseudo inverse?

Wiener Filter

This filter takes into account 1st and 2nd order statistics of the noise and image to generate the restoration filter transfer function. Additionally, it provides the best linear estimate of the image based up minimizing MSE (i.e. MMSE). However, it assumes wide-sense stationarity of the image field. We begin with the 1-D case.

1-D Wiener Filter:

Goal: Find $g(n)$'s or FIR filter coefficients so that $z(n)$ is as close as possible to the desired signal $d(n)$ by minimizing MSE $J(g) = E[e^2(n)]$.

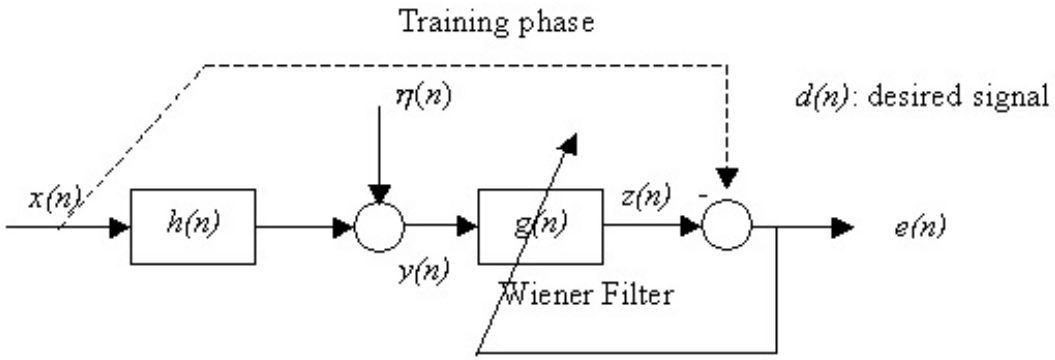


Figure 2: Wiener Filtering Process.

The MSE is given by

$$\begin{aligned} J(g(n)) &= E[e^2(n)] \\ with \;\; e(n) &= d(n) - z(n) \\ z(n) &= g(n) * y(n) = \sum_{i=0}^{N} g(i)y(n-i) \end{aligned}$$

Then

$$J(g) = E[(d(n) - \sum_{i=0}^{N} g(i)y(n-i))^2]$$

To find the optimum $g(k)$'s,

$$\frac{\partial J(g)}{\partial g(k)} = 0 \Longrightarrow -2E[y(n-k)(d(n) - \sum_{i=0}^{N} g(i)y(n-i))] = 0, \quad k \in [0, N]$$

Thus, we get

$$\sum_{i=0}^{N} g(i)E[y(n-i)y(n-k)] = E[d(n)y(n-k)]$$

or

$$\sum_{i=0}^{N} g(i) r_{yy}(k-i) = r_{dy}(k)$$

$$g(k) * r_{yy}(k) = r_{dy}(k)$$

where $r_{dy}(k) \triangleq E[d(n)y(n-k)]$ is the cross-correlation between $d(n)$ and $y(n)$ and $r_{yy}(k-i) \triangleq E[y(n-i)y(n-k)]$ is the auto-correlation of $y(n)$.

Taking the DTFT gives the general expression for the Wiener filter transfer function

$$G(e^{j\Omega}) = \frac{S_{dy}(e^{j\Omega})}{S_{yy}(e^{j\Omega})}$$

where

$$S_{dy}(e^{j\Omega}) = \text{DTFT}\{r_{dy}(k)\} \qquad \text{Cross} - \text{power spectrum}$$

$$S_{yy}(e^{j\Omega}) = \text{DTFT}\{r_{yy}(k)\} \qquad \text{Power spectrum}$$

Now, let $d(n) = x(n)$ (during the training phase) then assuming that the noise and signal are uncorrelated

$$\begin{aligned} y(n) &= h(n) * x(n) + \eta(n) \\ r_{yy}(k) &= y(k) * y(-k) \\ &= (h(k) * x(k) + \eta(k))(h(-k) * x(-k) + \eta(-k)) \\ &= h(k) * h(-k) * r_{xx}(k) + r_{\eta\eta}(k) \\ r_{dy}(k) &= d(k) * y(-k) \\ &= x(k) * (h(-k) * x(-k) + \eta(-k)) \\ &= h(-k) * r_{xx}(k) \end{aligned}$$

Thus, we get

$$\begin{aligned} S_{yy}(e^{j\Omega}) &= |H(e^{j\Omega})|^2 S_{xx}(e^{j\Omega}) + S_{\eta\eta}(e^{j\Omega}) \\ S_{dy}(e^{j\Omega}) &= H^*(e^{j\Omega}) S_{xx}(e^{j\Omega}) \end{aligned}$$

Hence leading to Wiener filter transfer function,

$$G(e^{j\Omega}) = \frac{H^*(e^{j\Omega}) S_{xx}(e^{j\Omega})}{|H(e^{j\Omega})|^2 S_{xx}(e^{j\Omega}) + S_{\eta\eta}(e^{j\Omega})}$$

Important Remarks

1. Once the transfer function of the Wiener filter is designed, the filtering can simply be done as shown in Fig. 3.

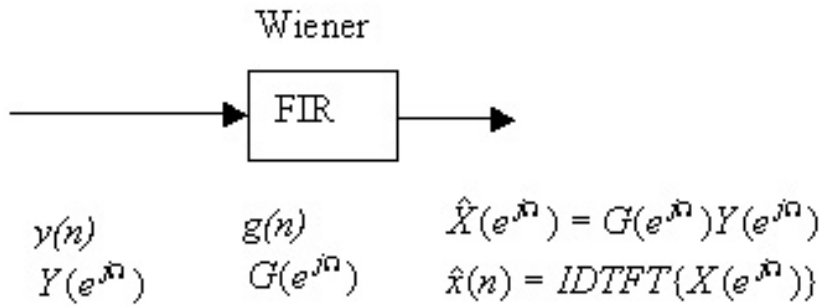


Figure 3: Wiener Filter Transfer Function.

2. If $h(n) = \delta(n) \rightarrow H(e^{j\Omega}) = 1$ i.e. signal plus noise $y(n) = x(n) + \eta(n)$, the transfer function simplifies to

$$G(e^{j\Omega}) = \frac{S_{xx}(e^{j\Omega})}{S_{xx}(e^{j\Omega}) + S_{\eta\eta}(e^{j\Omega})}$$

In this case, the power spectrum of the noise and signal can be measured from the power spectrum of the observed signal as shown in Fig. 4 (note that for zero mean white Gaussian noise $S_{\eta\eta}(e^{j\Omega}) = {\sigma_\eta}^2$).

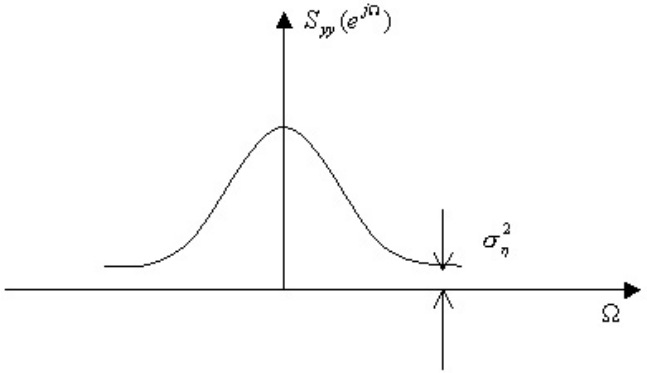


Figure 4: Estimating Noise Power.

3. A more useful form of the Wiener filter for frequency domain implementation is to work with the DFT instead of DTFT, in which case the transfer function becomes,

$$G(k) = \frac{H^*(k)S_{xx}(k)}{|H(k)|^2 S_{xx}(k) + S_{\eta\eta}(k)}$$

4. The general Wiener filter transfer function,

$$G(e^{j\Omega}) = \frac{S_{dy}(e^{j\Omega})}{S_{yy}(e^{j\Omega})} \qquad or \quad G(k) = \frac{S_{dy}(k)}{S_{yy}(k)}$$

can be applied to other types of observation models e.g., multiplicative noise case (see next example).

Example 2: Derive the transfer function of the Wiener filter for the following observation model.

$$y(n) = \gamma(n)[h(n) * x(n)] + \eta(n)$$

where $\gamma(n)$ and $\eta(n)$ are uncorrelated multiplicative and additive random noise with mean and correlations μ_γ and μ_η, and $r_{\gamma\gamma}(k)$ and $r_{\eta\eta}(k)$, respectively.

Solution: We use the result in Remark 4 to obtain $G(e^{j\Omega})$. Since $\gamma(n)$, $\eta(n)$ and $x(n)$ are mutually independent, using the convolution in frequency domain, the auto-correlation and power spectrum of the observed signal becomes

$$\begin{aligned} r_{yy}(k) &= y(k) * y(-k) \\ &= r_{\gamma\gamma}(k)[h(k) * h(-k) * r_{xx}(k)] + r_{\eta\eta}(k) \\ \text{and} \quad S_{yy}(e^{j\Omega}) &= S_{\gamma\gamma}(e^{j\Omega}) * *[|H(e^{j\Omega})|^2 S_{xx}(e^{j\Omega})] + S_{\eta\eta}(e^{j\Omega}) \end{aligned}$$

On the other hand, cross-correlation and cross-power spectrum for $d(k) = x(k)$ become

$$\begin{aligned} r_{dy}(k) &= d(k) * y(-k) \\ &= \mu_\gamma r_{xx}(k) * h(-k) \\ S_{dy}(e^{j\Omega}) &= \mu_\gamma S_{xx}(e^{j\Omega}) H^*(e^{j\Omega}) \end{aligned}$$

Thus, the transfer function becomes

$$G(e^{j\Omega}) = \frac{\mu_\gamma H^*(e^{j\Omega}) S_{xx}(e^{j\Omega})}{S_{\gamma\gamma}(e^{j\Omega}) * *[|H(e^{j\Omega})|^2 S_{xx}(e^{j\Omega})] + S_{\eta\eta}(e^{j\Omega})}$$

2-D Wiener Filter:

2-D extension is straightforward leading to 2-D Wiener transfer function

$$G(k,l) = \frac{H^*(k,l) S_{xx}(k,l)}{|H(k,l)|^2 S_{xx}(k,l) + S_{\eta\eta}(k,l)}$$

where $H(k,l) = 2-D\ DFT\{h(m,n)\}$, and $S_{xx}(k,l)$ and $S_{\eta\eta}(k,l)$ are power spectra of the original image $x(m,n)$ and additive noise, $\eta(m,n)$, respectively. The process is depicted in Fig.5.

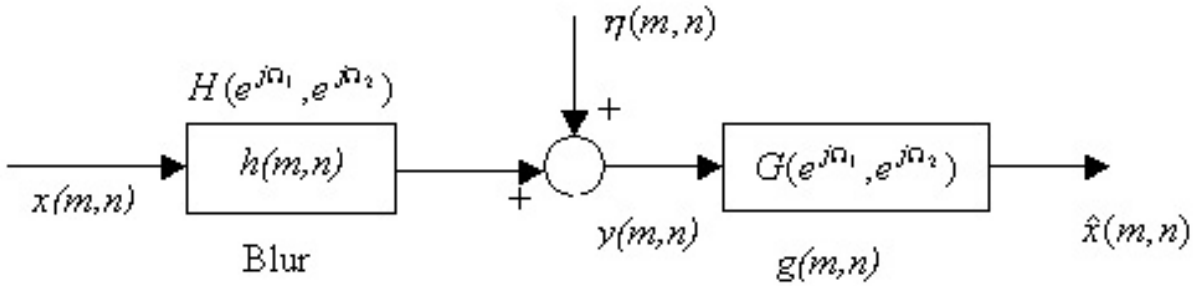


Figure 5: Image formation system and 2-D Wiener filtering.

Important Remarks

1. The above transfer function can be written as

$$G(k,l) = \frac{H^*(k,l)}{|H(k,l)|^2 + S_{\eta\eta}(k,l)/S_{xx}(k,l)}$$

where $S_{\eta\eta}(k,l)/S_{xx}(k,l) > 0$ is like ε in the pseudo inverse filter, but dependent on SNR. Thus, the Wiener filter does not have the singularity problem of the inverse filter.

2. If $S_{\eta\eta} >> S_{xx}$, $G(k,l)$ reduces to de-emphasize the noisy observation at that frequency. Recall that we have the opposite effects in the inverse filter, i.e. Wiener filter does not have ill-conditioning problems.

3. Assume no blur case, $h(m,n) = \delta(m,n) \rightarrow H(k,l) = 1$, then

$$G(k,l) = \frac{S_{xx}(k,l)}{S_{xx}(k,l) + S_{\eta\eta}(k,l)} = \frac{\mathrm{SNR}(k,l)}{\mathrm{SNR}(k,l)+1}$$

where

$$\mathrm{SNR}(k,l) \triangleq \frac{S_{xx}(k,l)}{S_{\eta\eta}(k,l)}$$

If $\mathrm{SNR} >> 1 \rightarrow G(k,l) \approx 1$, i.e. all frequency components are in the passband, while $\mathrm{SNR} << 1 \rightarrow G(k,l) \approx \mathrm{SNR}(k,l)$, i.e. all such frequencies are attenuated in proportion to their SNR values. This shows the adaptive nature of the Wiener filter.

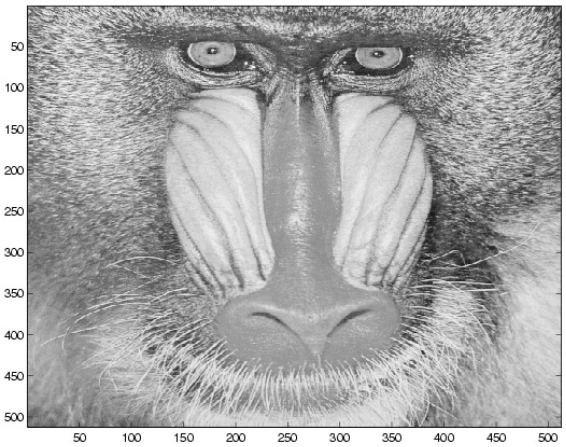
4. Since auto-correlation is real and even, S_{xx} and $S_{\eta\eta}$ are also real and non-negative. Thus, the phase of $G(k,l)$ is

$$\Phi_G = \Phi_{H^*} = -\Phi_H$$

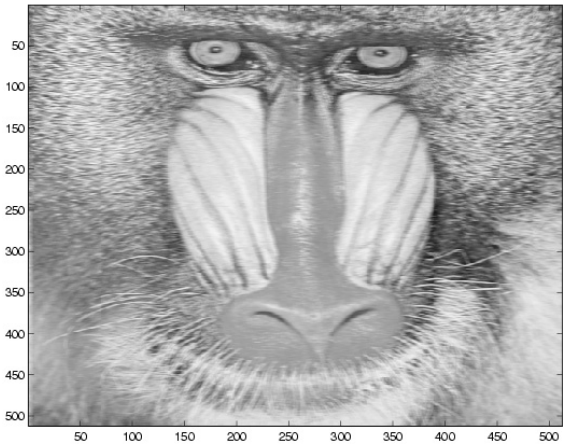
i.e. the phase of the inverse filter.

5. The main drawback of Wiener filter is the assumption of WSS for both image and noise. This is not obviously true for images. As result, the filtered images exhibit some edge smearing artifacts.

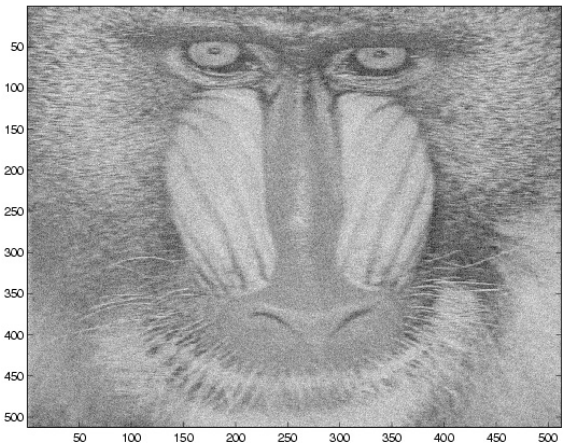
Figs. 6(a)-(d) show original, blurred (motion blur size 5), blurred and noisy (SNR=7 dB), and finally the Wiener filtered Baboon. Note: even some of the whisker details are restored.



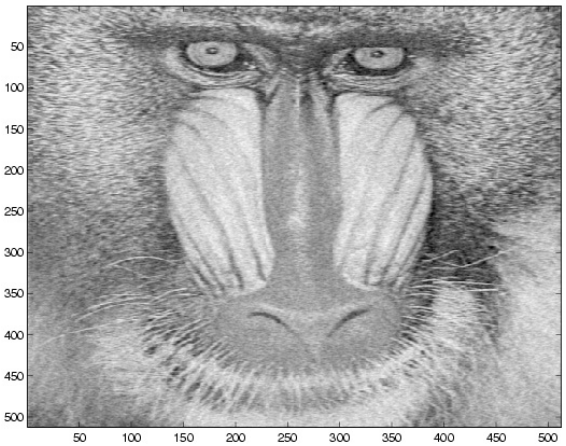
Original Image of Baboon



Baboon with Blurring Applied



Baboon with Blurring and Noise



Wiener Filter Result

Figure 6: Blurred and Noisy Baboon and Wiener Filter Results.

Recursive Restoration & Kalman Filtering

Wiener filter assumes WSS of the image field which leads to smearing of the edges and subtle textural features. This can be circumvented by recursive Kalman Filter (KF). KF requires:

1. 1-D or 2-D AR/ARMA model representation of the image field and covariance of the noise.
2. 1-D or 2-D state and observation equations with states being image pixels to estimate.
3. Recursive implementation of the filtering equations.

Remark: For WSS cases, KF becomes the same as Wiener filter.

Here, we discuss one type of recursive image restoration using *Strip Kalman Filter* (Azimi-Sadjadi, IEEE Trans. CAS, June 1989).

Strip Kalman Filter

Consider an image which is vector scanned horizontally in strips of width W. Image is assumed to be represented by an Mth order vector AR process with causal ROS as shown. Then,

$$\mathbf{z}(k) = A_1^T \mathbf{z}(k-1) + \cdots + A_M^T \mathbf{z}(k-M) + \mathbf{e}(k)$$

where $\mathbf{z}(k)$ is a vector of pixels of size W, $\mathbf{e}(k)$ is the driving noise vector and A_is are model coefficient matrices of size $W \times W$. We assume that $E[\mathbf{e}(k)] = 0$ and $E[\mathbf{e}(k)\mathbf{e}^T(j)] = R_{\mathbf{e}}\delta(k-j)$ with $R_{\mathbf{e}}$ being the covariance matrix of $\mathbf{e}(k)$.

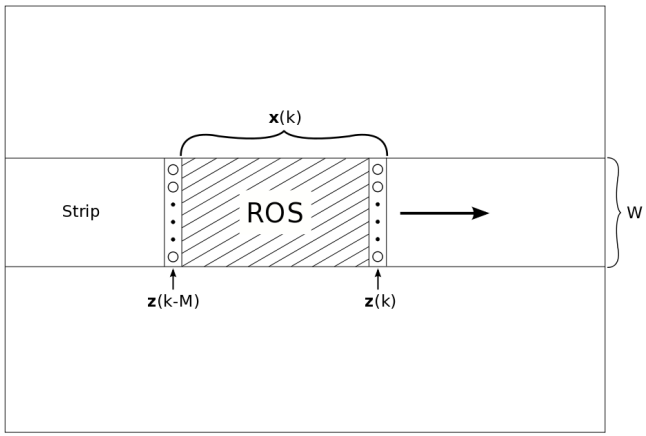


Figure 7: Strip Kalman Filtering.

Note that the model coefficient matrices can be identified using vector Yule-Walker equation. To see this let's form covariance matrix of data from the vector AR model above. This gives

$$R_{\mathbf{z}}(m) = E[\mathbf{z}(k)\mathbf{z}^T(k-m)] = R_{\mathbf{z}}(m-1)A_1 + \cdots + R_{\mathbf{z}}(m-M)A_M + R_{\mathbf{e}}\delta(m)$$

which yields

$$\begin{bmatrix} R_{\mathbf{z}}(0) & R_{\mathbf{z}}^T(1) & \cdots & \cdots & R_{\mathbf{z}}^T(M) \\ R_{\mathbf{z}}(1) & R_{\mathbf{z}}(0) & \cdots & \cdots & R_{\mathbf{z}}^T(M-1) \\ \vdots & & & & \vdots \\ & & \ddots & & \\ \vdots & & & \ddots & \vdots \\ R_{\mathbf{z}}(M) & R_{\mathbf{z}}(M-1) & \cdots & \cdots & R_{\mathbf{z}}(0) \end{bmatrix} \begin{bmatrix} I_W \\ -A_1 \\ \vdots \\ \\ \vdots \\ -A_M \end{bmatrix} = \begin{bmatrix} R_{\mathbf{e}} \\ \mathbf{0} \\ \mathbf{0} \\ \\ \vdots \\ \mathbf{0} \end{bmatrix}$$

Here $R_{\mathbf{z}}(-m) = R_{\mathbf{z}}^T(m)$. Solving this vector Yule Walker equation leads to the parameter matrices for AR model. The sample data covariance matrices $\hat{R}_{\mathbf{z}}(m)$ in this equation are estimated using the training images and $\hat{R}_{\mathbf{z}}(m) = \frac{1}{N}\sum_{k=0}^{N-1}\mathbf{z}(k)\mathbf{z}^T(k-m)$.

Next, we form the state equation by defining the *state* vector (size $WM \times 1$) as

$$\mathbf{x}(k) = [\mathbf{z}^T(k)\mathbf{z}^T(k-1)\cdots\mathbf{z}^T(k-M+1)]^T$$

then *state equation* (in canonical form) is

$$\mathbf{x}(k) = F\mathbf{x}(k-1) + G\mathbf{e}(k)$$

where

$$F = \begin{bmatrix} A_1^T & A_2^T & \cdots & \cdots & A_M^T \\ I_W & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \vdots & & & & \vdots \\ & & \ddots & & \\ \vdots & & & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \cdots & I_W \end{bmatrix}, \quad G = \begin{bmatrix} I_W \\ \mathbf{0} \\ \vdots \\ \\ \vdots \\ \mathbf{0} \end{bmatrix}$$

Now, consider vector observation equation in which blur is modeled as an MA (or FIR) process with ROS limited to that of AR model, i.e.

$$\mathbf{y}(k) = H_0\mathbf{z}(k) + H_1\mathbf{z}(k-1) + \cdots + H_{M-1}\mathbf{z}(k-M+1) + \mathbf{n}(k)$$

where $\mathbf{y}(k)$ is the observation vector and $\mathbf{n}(k)$ represents the additive noise vector with $E[\mathbf{n}(k)] = 0$ and $E[\mathbf{n}(k)\mathbf{n}^T(j)] = R_{\mathbf{n}}\delta(k-j)$ with $R_{\mathbf{n}}$ being the covariance matrix of $\mathbf{n}(k)$. In vector form, the observation equation becomes

$$\mathbf{y}(k) = H\mathbf{x}(k) + \mathbf{n}(k)$$

where $H = [H_0 \cdots H_{M-1}]$.

The aim of Kalman filter is to generate estimates of the state vector (image) recursively based upon all available observations i.e. $\hat{\mathbf{x}}(k|k) = E[\mathbf{x}(k)|Z(k)]$ where $Z(k) = \{\mathbf{z}(1), \cdots, \mathbf{z}(k)\}$ is a set of all observations up to time k. Then, the Kalman filter equations in order are:

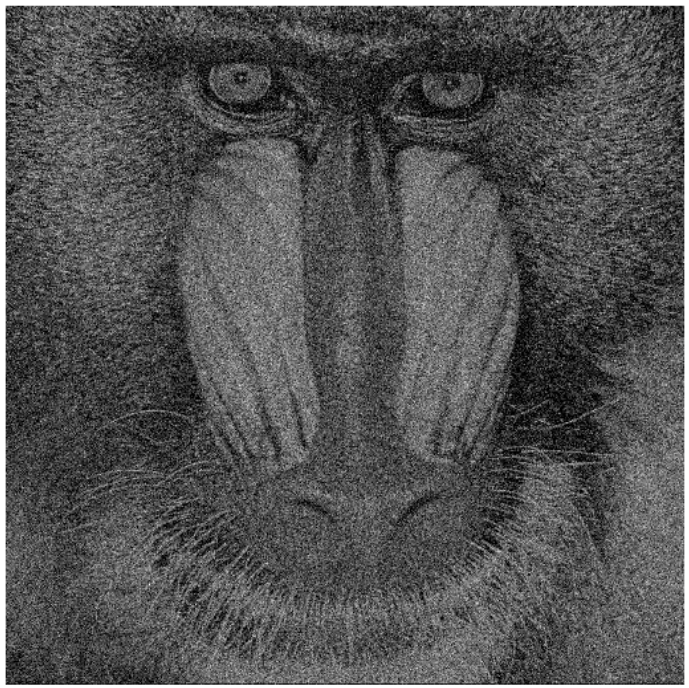
1. *Initial Estimate:* $\hat{\mathbf{x}}(k|k-1) = F\hat{\mathbf{x}}(k-1|k-1)$.
2. *A priori Error Covariance Matrix:*
   $P(k|k-1) = FP(k-1|k-1)F^T + GR_{\mathbf{e}}G^T$ where
   $P(k|k-1) = E[(\mathbf{x}(k) - \hat{\mathbf{x}}(k|k-1))(\mathbf{x}(k) - \hat{\mathbf{x}}(k|k-1))^T]$.
3. *Kalman Gain:* $K(k) = P(k|k-1)H^T[HP(k|k-1)H^T + R_{\mathbf{n}}]^{-1}$.

4. *Update Estimate:* $\hat{\mathbf{x}}(k|k) = \hat{\mathbf{x}}(k|k-1) + K(k)[\mathbf{y}(k) - H\hat{\mathbf{x}}(k|k-1)]$.
5. *A posteriori Error Covariance Matrix:*
   $P(k|k) = [I - K(k)H]P(k|k-1)$ where
   $P(k|k) = E[(\mathbf{x}(k) - \hat{\mathbf{x}}(k|k))(\mathbf{x}(k) - \hat{\mathbf{x}}(k|k))^T]$.

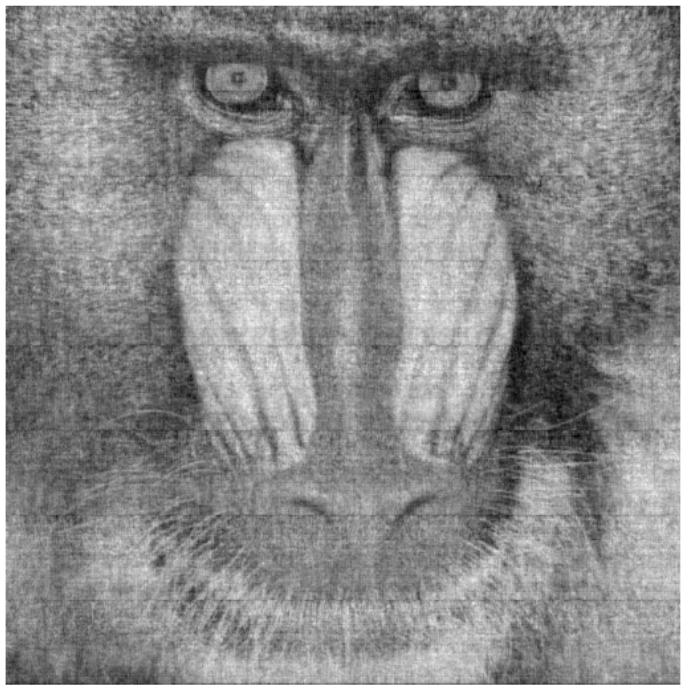
Important Remarks:

1. Initial conditions are $\hat{\mathbf{x}}(0|0) = \mu_x[1, \cdots, 1]^T$ and $P(0|0) = P_x(0)$ which is a measure of confidence in μ_x i.e. larger $P_x(0)$ for lesser confidence.
2. KF is driven by the *innovation process* $\boldsymbol{\xi}(k) = \mathbf{y}(k) - H\hat{\mathbf{x}}(k|k-1)$.
3. Eqs. in steps 2, 3, and 5 can be iterated off-line until convergence is reached for $K(k)$ (stationary case) and then $K(k)$ is used on-line in the state updating process.
4. Note that Kalman gain matrix decreases when either effects of $R_{\mathbf{e}}$ or $P_x(0)$ decrease or when $R_{\mathbf{n}}$ increases. When $R_{\mathbf{e}} \approx \mathbf{0}$ then $K(k)$ asymptotically approaches to zero and hence KF becomes independent of observations (data saturation may occur).

Figs. 8(a) and (b) show the noisy (additive WGN) Baboon with SNR_{input} =-1.7 dB and strip KF processed Baboon with SNR_{output} =2.83 dB (i.e. >4.5dB improvements). Vector AR(1) was used with $W = 64$. Clearly, the results are impressive.



Noisy Baboon



KF Processed Baboon.

Figure 8: Noisy Baboon and Strip Kalman Filter Results.