UNIVERSITY OF MINNESOTA
Driven to Discover℠

EE 5561: Image Processing and Applications

Lecture 9

Mehmet Akçakaya

Recap of Last Lecture

- Linear algebra review
  - Wavelet transforms via linear algebra
    - Haar wavelets, unitary
  - Energy compaction
  - Another transform with good energy compaction (DCT)
  - Random variables & vectors review

- Today
  - Start on image restoration

Enhancement vs. Restoration

- Enhancement
  - Pleasing images
  - No model for degradation
  - Ad hoc procedures
  - Visual metric (mostly)

- Restoration
  - “Inverting” a degradation
  - Model exists
  - Quantitative evaluation

Enhancement vs. Restoration

- Degradations are common in imaging systems
  - Either by design or physical limitations

- Restoration relies on optimization procedures
  - We already saw least squares as a basic example

- Related to math field “inverse problems”

- Sometimes also called “computational imaging”

Image Denoising

- First problem we will consider in this module

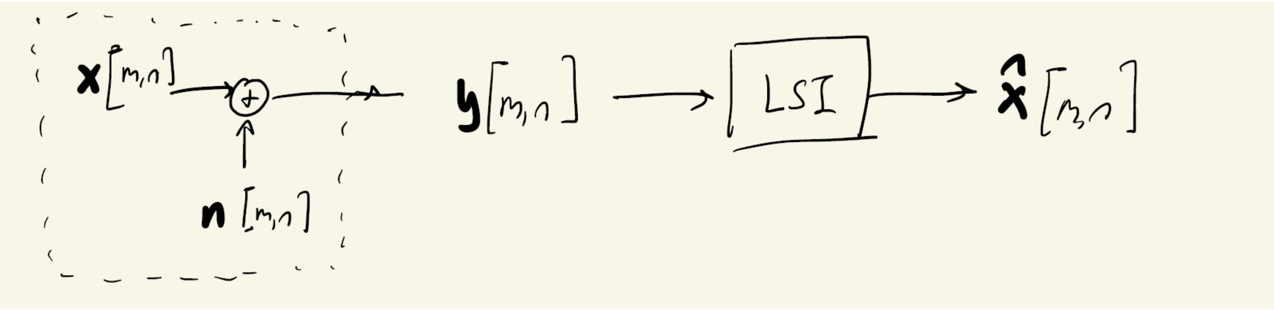
$$\mathbf{y} = \mathbf{x} + \mathbf{n}$$

observed image (in vector form)

true but unknown underlying image

additive random noise

  - Goal: Recover **x** from **y**
  - Our first attempt will be based on Wiener filter
    - Original version (with random processes) by Wiener in 1930s
    - Easy to implement a linear shift invariant filter for recovery



The original formulation considers signals of possibly infinite extent (not vectors)

Image Denoising – Wiener Filter

- In matrix form we will want

$$\hat{\mathbf{x}} = \mathbf{G}\mathbf{y}$$

(just a linear system)

- Wiener's criterion: Minimize the average mean squared error
  - **x** itself is an instance of a random vector **X̲**
  - I am going to drop the notation for random vectors, and use them interchangeably, since the context is clear here
  - i.e. minimize $\mathbb{E}\left[||\mathbf{x} - \hat{\mathbf{x}}||_2^2\right]$

Image Denoising – Wiener Filter

- We have

$$\begin{aligned}\epsilon^2 = \mathbb{E}\left[||\mathbf{x} - \hat{\mathbf{x}}||_2^2\right] &= \mathbb{E}\left[(\mathbf{x} - \hat{\mathbf{x}})^H (\mathbf{x} - \hat{\mathbf{x}})\right] \\ &= \mathbb{E}\left[tr\left((\mathbf{x} - \hat{\mathbf{x}})^H (\mathbf{x} - \hat{\mathbf{x}})\right)\right] \\ &= \mathbb{E}\left[tr\left((\mathbf{x} - \hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}})^H\right)\right] \\ &= tr\left(\mathbb{E}\left[(\mathbf{x} - \hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}})^H\right]\right) \\ &= tr\left(\mathbb{E}\left[\mathbf{x}\mathbf{x}^H - \hat{\mathbf{x}}\mathbf{x}^H - \mathbf{x}\hat{\mathbf{x}}^H + \hat{\mathbf{x}}\hat{\mathbf{x}}^H\right]\right)\end{aligned}$$

tr(**A**): sum of diagonal elements of square matrix **A**

since tr(**AB**) = tr(**BA**)

since tr(·) and $\mathbb{E}(\cdot)$ are linear

- We need other assumptions
  - **x** and **n** are uncorrelated (important)
  - **n** is zero-mean (can be worked around but for easier notation)

Image Denoising – Wiener Filter

- Then

$$\epsilon^2 = tr\left(\mathbb{E}\left[\mathbf{x}\mathbf{x}^H - \hat{\mathbf{x}}\mathbf{x}^H - \mathbf{x}\hat{\mathbf{x}}^H + \hat{\mathbf{x}}\hat{\mathbf{x}}^H\right]\right)$$

  - Here $\mathbb{E}\left[\mathbf{x}\mathbf{x}^H\right] = \mathbf{R}_\mathbf{x}$

$$\begin{aligned}
\mathbb{E}\left[\hat{\mathbf{x}}\mathbf{x}^H\right] = \mathbb{E}\left[(\mathbf{G}\mathbf{y})\mathbf{x}^H\right] &= \mathbb{E}\left[\mathbf{G}(\mathbf{x}+\mathbf{n})\mathbf{x}^H\right] \\
&= \mathbb{E}\left[\mathbf{G}\mathbf{x}\,\mathbf{x}^H + \mathbf{G}\mathbf{n}\,\mathbf{x}^H\right] \\
&= \mathbf{G}\left(\mathbb{E}\left[\mathbf{x}\,\mathbf{x}^H + \mathbf{n}\,\mathbf{x}^H\right]\right) \\
&= \mathbf{G}\left(\mathbb{E}\left[\mathbf{x}\,\mathbf{x}^H\right] + \mathbb{E}\left[\mathbf{n}\right]\mathbb{E}\left[\mathbf{x}^H\right]\right) \\
&= \mathbf{G}\left(\mathbb{E}\left[\mathbf{x}\,\mathbf{x}^H\right]\right) = \mathbf{G}\mathbf{R}_\mathbf{x}
\end{aligned}$$

Image Denoising – Wiener Filter

- Then

$$\epsilon^2 = tr\Big(\mathbb{E}\big[\mathbf{x}\mathbf{x}^H - \hat{\mathbf{x}}\mathbf{x}^H - \mathbf{x}\hat{\mathbf{x}}^H + \hat{\mathbf{x}}\hat{\mathbf{x}}^H\big]\Big)$$

  - Here

$$\mathbb{E}\big[\mathbf{x}\mathbf{x}^H\big] = \mathbf{R_x}$$

$$\mathbb{E}\big[\hat{\mathbf{x}}\mathbf{x}^H\big] = \mathbf{G}\mathbf{R_x}$$

$$\begin{aligned}\mathbb{E}\big[\hat{\mathbf{x}}\hat{\mathbf{x}}^H\big] = \mathbb{E}\big[\mathbf{G}\mathbf{y}\mathbf{y}^H\mathbf{G}^H\big] &= \mathbf{G}\mathbb{E}\big[\mathbf{y}\mathbf{y}^H\big]\mathbf{G}^H \\ &= \mathbf{G}\mathbb{E}\big[(\mathbf{x}+\mathbf{n})(\mathbf{x}+\mathbf{n})^H\big]\mathbf{G}^H \\ &= \mathbf{G}\mathbb{E}\big[\mathbf{x}\mathbf{x}^H + \mathbf{n}\mathbf{x}^H + \mathbf{x}\mathbf{n}^H + \mathbf{n}\mathbf{n}^H\big]\mathbf{G}^H \\ &= \mathbf{G}(\mathbf{R_x} + \mathbf{R_n})\mathbf{G}^H\end{aligned}$$

  - So we have

$$\epsilon^2 = tr\Big(\mathbf{R_x} - \mathbf{G}\mathbf{R_x} - \mathbf{R_x}^H\mathbf{G}^H + \mathbf{G}\mathbf{R_x}\mathbf{G}^H + \mathbf{G}\mathbf{R_n}\mathbf{G}^H\Big)$$

Image Denoising – Wiener Filter

- We want to minimize this error with respect to **G**

$$\epsilon^2 = tr\left(\mathbf{R_x} - \mathbf{G}\mathbf{R_x} - \mathbf{R_x}^H\mathbf{G}^H + \mathbf{G}\mathbf{R_x}\mathbf{G}^H + \mathbf{G}\mathbf{R_n}\mathbf{G}^H\right)$$

  - i.e. we want to set $\frac{\partial \epsilon^2}{\partial \mathbf{G}} = 0$
  - How? $$\frac{\partial \epsilon^2}{\partial \mathbf{G}} = \begin{bmatrix} \frac{\partial \epsilon^2}{\partial g_{11}} & \frac{\partial \epsilon^2}{\partial g_{12}} & \cdots \\ \vdots & & \end{bmatrix}$$
  - This becomes trickier since the matrices can be complex

Image Denoising – Wiener Filter

- Complex matrix calculus (IEEE TSP, 55(6): 2740-46)

$$\frac{\partial}{\partial \mathbf{G}} tr(\mathbf{A}\mathbf{G}) = \mathbf{A}^T$$
$$\frac{\partial}{\partial \mathbf{G}} tr(\mathbf{G}^H\mathbf{A}) = \mathbf{0}$$
$$\frac{\partial}{\partial \mathbf{G}} tr(\mathbf{G}\mathbf{A}\mathbf{G}^H) = \mathbf{G}^H\mathbf{A}$$

$$\frac{\partial}{\partial \mathrm{conj}(\mathbf{G})} tr(\mathbf{A}\mathbf{G}) = \mathbf{0}$$
$$\frac{\partial}{\partial \mathrm{conj}(\mathbf{G})} tr(\mathbf{G}^H\mathbf{A}) = \mathbf{A}$$
$$\frac{\partial}{\partial \mathrm{conj}(\mathbf{G})} tr(\mathbf{G}\mathbf{A}\mathbf{G}^H) = \mathbf{G}\mathbf{A}$$

- Based on these easier to set

$$
\begin{aligned}
0 &= \frac{\partial \epsilon^2}{\partial \mathrm{conj}(\mathbf{G})} \\
&= \frac{\partial}{\partial \mathrm{conj}(\mathbf{G})} tr(\mathbf{R_x}) - \frac{\partial}{\partial \mathrm{conj}(\mathbf{G})} tr(\mathbf{G}\mathbf{R_x}) - \frac{\partial}{\partial \mathrm{conj}(\mathbf{G})} tr(\mathbf{R_x}^H\mathbf{G}^H) + \frac{\partial}{\partial \mathrm{conj}(\mathbf{G})} tr(\mathbf{G}(\mathbf{R_x}+\mathbf{R_n})\mathbf{G}^H) \\
&= \mathbf{0} - \mathbf{0} - \mathbf{R_x}^H + \mathbf{G}(\mathbf{R_x}+\mathbf{R_n}) \\
&= -\mathbf{R_x} + \mathbf{G}(\mathbf{R_x}+\mathbf{R_n})
\end{aligned}
$$

Image Denoising – Wiener Filter

- We want to minimize this error with respect to **G**

$$\epsilon^2 = tr\left(\mathbf{R_x} - \mathbf{G}\mathbf{R_x} - \mathbf{R_x}^H\mathbf{G}^H + \mathbf{G}\mathbf{R_x}\mathbf{G}^H + \mathbf{G}\mathbf{R_n}\mathbf{G}^H\right)$$

  - Leads to $\mathbf{G} = \mathbf{R_x}(\mathbf{R_x} + \mathbf{R_n})^{-1}$

  - Assuming white noise gives $\mathbf{R_n} = \sigma^2\mathbf{I}$

  - $\mathbf{R_x}$ is Hermitian positive semi-definite (it is a correlation matrix) → diagonalize it

$$\mathbf{R_x} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^H \qquad \text{with } \mathbf{U} = [\mathbf{u}_1 \ldots \mathbf{u}_n] \text{ unitary}$$

$$\text{and } \Lambda_{ii} = \lambda_i \text{ diagonal}$$

Image Denoising – Wiener Filter

– We want to minimize this error with respect to **G**

$$\epsilon^2 = tr\Big(\mathbf{R_x} - \mathbf{G}\mathbf{R_x} - \mathbf{R_x}^H\mathbf{G}^H + \mathbf{G}\mathbf{R_x}\mathbf{G}^H + \mathbf{G}\mathbf{R_n}\mathbf{G}^H\Big)$$

- With these

$$\begin{aligned}
\mathbf{G} &= \mathbf{R_x}(\mathbf{R_x} + \mathbf{R_n})^{-1} \\
&= \mathbf{U}\mathbf{\Lambda}\mathbf{U}^H(\mathbf{U}\mathbf{\Lambda}\mathbf{U}^H + \sigma^2\mathbf{I})^{-1} \\
&= \mathbf{U}\mathbf{\Lambda}\mathbf{U}^H\Big(\mathbf{U}\big(\mathbf{\Lambda} + \sigma^2\mathbf{I}\big)\mathbf{U}^H\Big)^{-1} \\
&= \mathbf{U}\mathbf{\Lambda}\big(\mathbf{\Lambda} + \sigma^2\mathbf{I}\big)^{-1}\mathbf{U}^H \\
&= \sum_{k=1}^{n} \frac{\lambda_k}{\lambda_k + \sigma^2}\mathbf{u}_k\mathbf{u}_k^H
\end{aligned}$$

Image Denoising – Wiener Filter

- How does this work in reality?
  - $\mathbf{R_x}$ is generally not available (or equivalently $\mathbf{U}$ and $\mathbf{\Lambda}$ are not available)
  - Need to estimate it
  - One approach[1] is to approximate $\mathbf{U}$ with a pre-determined transform (e.g. wavelet), then estimate $\mathbf{\Lambda}$ from transform coefficients

$$\lambda_i = \begin{cases} |\theta(i)|^2 & \text{for } i = 1, \ldots, n_s \qquad \text{(based on some threshold for energy preservation)} \\ 0 & \text{otherwise} \end{cases}$$

  - Alternative: For large n, $\mathbf{R_x}$ diganolized by IDFT (~ random process limit)
  - This ties in with how for wide-sense stationary random processes, spectral density is the Fourier transform of correlation function

[1]Ghael et al, SPIE, 1997

Image Denoising – Wiener Filter

- How does this work in reality?
  - Or patch processing
  - We already mentioned that block processing is popular in image processing (and we'll see more)
    - Define a small neighborhood
    - Assume similar signal content here
    - In this case, this means: image pixels in small neighborhood come from the same distribution
    - Calculate sample mean, standard deviation
    - For pixels $\{y_1, \ldots, y_n\}$

$$\hat{\mu}_y = \frac{1}{n}\sum_{k=1}^{n} y_k$$

$$\hat{\sigma}_y = \sqrt{\frac{\sum_{k=1}^{n}(y_k - \hat{\mu}_y)^2}{n-1}}$$

Bessel's correction

Performance Measures

- We saw our first restoration tool, designed to minimize a certain "loss" based on a statistical model
  - We modeled both noise and image as instances of random vectors
  - Used an expected l_2 loss
- We will see such losses or performance measures for the remainder of the course
  - Here we'll briefly review some common ones

Performance Measures

- Mean squared error (MSE):

$$\mathrm{MSE} = \frac{1}{N}||\mathbf{x} - \hat{\mathbf{x}}||_2^2 \qquad \mathbf{x} \in \mathbb{R}^N \ \text{(vectorized image)}$$

or

$$\mathrm{MSE} = \mathbb{E}\left[||\mathbf{x} - \hat{\mathbf{x}}||_2^2\right] \qquad \text{if } \mathbf{x} \text{ is a random vector}$$

  - This is per-pixel average error
  - Its units: square of the original image unit
  - We sometimes want the same units as the image, especially in quantitative imaging modalities (e.g. CT)

- Root mean squared error $\mathrm{RMSE} = \sqrt{\mathrm{MSE}}$

Performance Measures

- Normalized MSE (NMSE)
  - MSE and RMSE both depend on how **x** is scaled (e.g. is it between 0-1 or 0-255?)
  - NMSE tackles this

$$\mathrm{NMSE} = \frac{\mathrm{MSE}}{\frac{1}{N}||\mathbf{x}||_2^2} = \frac{||\mathbf{x}-\hat{\mathbf{x}}||_2^2}{||\mathbf{x}||_2^2}$$

or statistical version

$$\mathrm{NMSE} = \frac{\mathrm{MSE}}{\mathbb{E}\left[||\mathbf{x}||_2^2\right]}$$

- SNR = $\frac{1}{\mathrm{NMSE}}$ or $10\log_{10}\frac{1}{\mathrm{NMSE}}$

Performance Measures

– Peak SNR (PSNR)

$$\mathrm{PSNR} = 10\log_{10}\frac{\max_k |x_k|^2}{\mathrm{MSE}}$$

- Very common for evaluating image quality

– Worst case (l_∞)

$$\max_k |\hat{x}_k - x_k| = ||\mathbf{x} - \hat{\mathbf{x}}||_\infty$$

– l_1 error

$$\frac{1}{N}||\mathbf{x} - \hat{\mathbf{x}}||_1 = \frac{1}{N}\sum_{k=1}^{N}|\hat{x}_k - x_k|$$

Performance Measures

− Structural Similarity Index (SSIM)

- Main idea: Previous metrics work on quantifying point-wise differences between images. But our perception system identifies structural information & differences in those.
- Need a metric to capture this.
- Relies on 3 features: luminance, contrast, structure
- Luminance: Averaging over all pixels. For image **x**, $\mu_x = \frac{1}{N}\sum_{k=1}^{N} x_k$
- Contrast: Standard deviation across pixels. $\sigma_x = \left(\frac{1}{N-1}\sum_{k=1}^{N}(x_k - \mu_x)^2\right)^{\frac{1}{2}}$
- Structure: Normalized input signal $(\mathbf{x} - \mu_x)/\sigma_x$

- Then we build functions to compare each of these between two given images…
- And combine these information
- After some work, the standard formula is $\mathrm{SSIM}(\mathbf{x},\mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$

Wang et al, IEEE TIP, 2004

Performance Measures

- Structural Similarity Index (SSIM)
  - This is a global metric
  - It is better to apply the metrics regionally (on patches), and then average these
  - Technically called mean structural similarity index
  - Leads to weighted averages (e.g. with a Gaussian weighting) instead of simple averaging for luminance, contrast and structure
  - Final result is the average across all patches

Performance Measures

- Not covered now
  - Perceptual losses

- These and SSIM will be important later in the ML/AI module too