University of Minnesota
Driven to Discover℠

EE 5561: Image Processing and Applications

Lecture 4

Mehmet Akçakaya

Last Lecture Recap

- Fourier analysis review
  - Fourier series for periodic (or finite-extent) images
  - Fourier transform as a more general tool
  - How Fourier analysis describes images
    - Low-frequency content → contrast information, most of the energy of the image
    - High-frequency content → edges and other sharper structures
- Sampling & interpolation
  - The tools that allow us to move to discrete-space
- 2D discrete-space signals introduction

2D Discrete-Space Systems

$$f[m,n] \longrightarrow \boxed{\mathbf{S}} \longrightarrow g[m,n]$$

- Recall Kronecker delta

$$\delta[m,n] = \begin{cases} 1 & \text{if } m = n = 0 \\ 0 & \text{otherwise} \end{cases}$$

- Properties analogous to Dirac delta for continuous-space signals

$$\sum_{m,n} \delta[m,n] = 1$$

$$\delta\left[m - m_0, n - n_0\right] \cdot f\left[m,n\right] = f\left[m_0, n_0\right] \cdot \delta\left[m - m_0, n - n_0\right]$$ sampling

$$\sum_{m,n} \left(\delta\left[m - m_0, n - n_0\right] \cdot f\left[m,n\right]\right) = f\left[m_0, n_0\right]$$ sifting

$$\delta\left[m,n\right] = \delta_x[m] \cdot \delta_y[n]$$ separable

2D Discrete-Space Systems

$$f[m,n] \longrightarrow \boxed{\textbf{S}} \longrightarrow g[m,n] = \mathcal{S}\big(f[m,n]\big)$$

- Example: moving average over a 3×3 window

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$g[m,n] = \frac{1}{9}\sum_{k=m-1}^{m+1}\sum_{l=n-1}^{n+1} f[k,l]$$

$$= \frac{1}{9}\sum_{k=-1}^{1}\sum_{l=-1}^{1} f[m-k,n-l]$$

- Example: segmentation based on a simple threshold

$$g[m,n] = \begin{cases} 1 & \big|f\left[m_n\right]\big| > T \\ 0 & \text{otherwise} \end{cases}$$

2D Discrete-Space Systems

$$f[m,n] \longrightarrow \boxed{\mathbf{S}} \longrightarrow g[m,n] = \mathcal{S}\big(f[m,n]\big)$$

- Example: downsampling (trivial compression)

$$g[m,n] = f[2m,2n]$$

- Example: upsampling

$$g[m,n] = \begin{cases} f[\frac{m}{2},\frac{n}{2}] & \text{m,n even} \\ 0 & \text{otherwise} \end{cases}$$

not ideal!

should interpolate…

2D Discrete-Space Systems

- Properties:
  - Amplitude-based: linearity, stability, invertibility
  - Spatial-based: causality, separability, shift-invariance,…
- Similar to continuous-space cases
  - Linearity

$$\mathcal{S}(\alpha f_1 + \beta f_2) = \alpha\mathcal{S}(f_1) + \beta\mathcal{S}(f_2)$$

    - e.g. thresholding (earlier) $f_1 + f_2 > T$ but $f_1 < T \text{ and } f_2 < T$ → non-linear
  - Stability: BIBO
    - What about? $g[m,n] = g[m-1, n-1] + f[m,n]$
    - Take $f[m,n] = 1$
    - Bounded input does not produce bounded output

2D Discrete-Space Systems

- Invertibility
  - Downsampling? No
  - Upsampling? Yes
- Shift-invariance

$$\text{If} \qquad f[m,n] \rightarrow \mathcal{S} \rightarrow g[m,n] \qquad \text{then}$$

$$f[m-m_0, n-n_0] \rightarrow \mathcal{S} \rightarrow g[m-m_0, n-n_0] \qquad \forall m_0, n_0 \in \mathbb{Z}, \forall f$$

  - Moving average filter?

$$g[m,n] = \mathcal{S}\big(f[m,n]\big) = \frac{1}{9}\sum_{k=-1}^{1}\sum_{l=-1}^{1} f[m-k, n-l]$$

$$\mathcal{S}\big(f[m-m_0, n-n_0]\big) = \frac{1}{9}\sum_{k=-1}^{1}\sum_{l=-1}^{1} f[m-m_0-k, n-n_0-l]$$

same!

$$g[m-m_0, n-n_0] = \frac{1}{9}\sum_{k=-1}^{1}\sum_{l=-1}^{1} f[m-m_0-k, n-n_0-l]$$

2D Discrete-Space LSI Systems

- Same idea as in continuous space
  - Instead using Kronecker delta as a building block
  - For a general system: $\delta[m,n] \rightarrow \mathcal{S} \rightarrow h[m,n]$
  - SI system: $\delta[m-k,n-l] \rightarrow \mathcal{S} \rightarrow h[m-k,n-l]$
  - LSI system: $f[m,n] \rightarrow \mathcal{S} \rightarrow g[m,n]$

$$f[m,n] = \sum_{k,l} f[k,l]\delta[m-k,n-l] \rightarrow \mathcal{S} \rightarrow \sum_{k,l} f[k,l]h[m-k,n-l] = g[m,n]$$

$$= f[m,n] * h[m,n]$$

discrete convolution

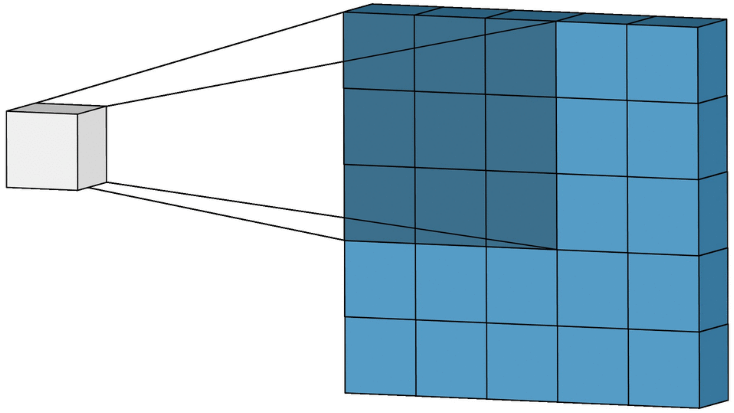
  - This convolution operation will be fundamental
    - Will be used from linear filters to neural networks

Convolution Sum

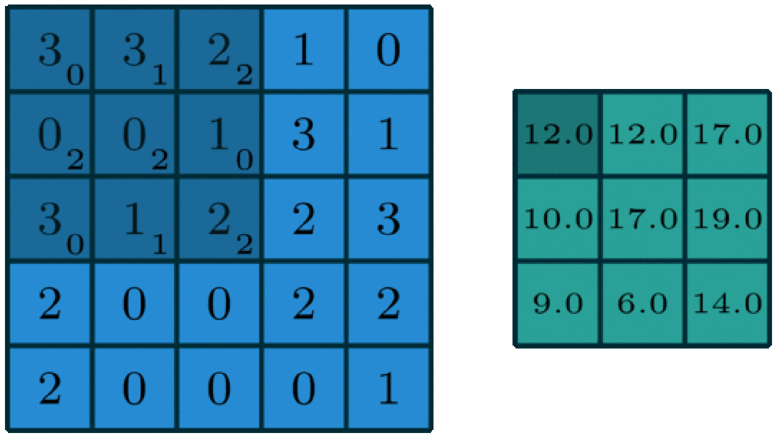
- How to calculate the convolution sum? $\sum_{k,l} f[k,l]h[m-k,n-l]$
  - In signals & systems, view f and h as functions of [k,l] for fixed [m,n]
  - Here $h[m-k,n-l]$ corresponds to a shifted & reversed version of $h[k,l]$
  - Process:
    - Fix m,n → Generate $h[m-k,n-l]$ with the appropriate shift & reversal
    - Multiply by $f[k,l]$ at each point → sum over all k,l
    - Repeat over all m,n
  - Usually implemented as a sliding window
  - This is a little complicated… Why reverse for each m,n? etc
- In practice, we just specify the “reversed” filter

Convolution Sum

- How to calculate the convolution sum?
  - Pictorially



  - An example:



Moving Average Filter Example

$f[m,n]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[m,n]$

Ref: Seitz

Moving Average Filter Example

$f[m, n]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[m, n]$

	0	10							

Ref: Seitz

Moving Average Filter Example

$f[m,n]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[m,n]$

	0	10	20						

Ref: Seitz

Moving Average Filter Example

$f[m,n]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[m,n]$

	0	10	20	30					

Ref: Seitz

Moving Average Filter Example

$f[m,n]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[m,n]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Ref: Seitz

Moving Average Filter Example

Original

3×3 kernel

5×5 kernel

Moving Average Filter Example

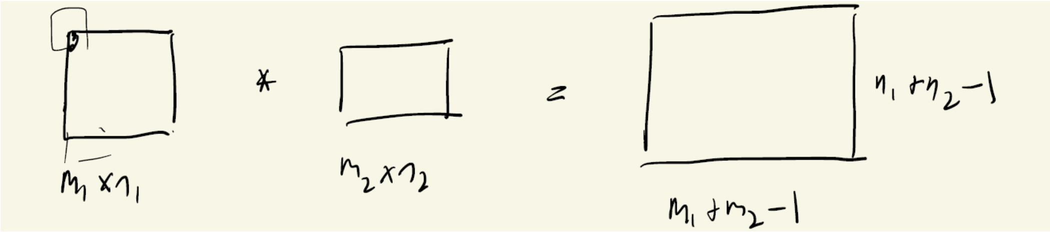
Original

Noisy

3×3 kernel

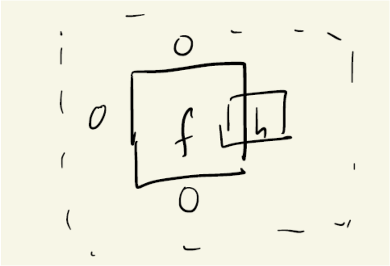
Convolution Sum

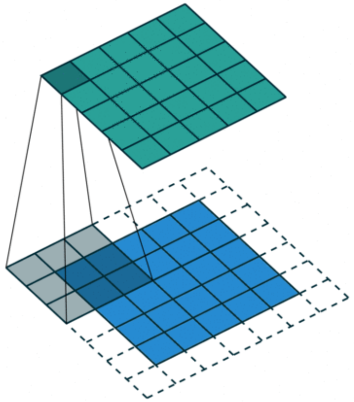
- Going back: What happens at the boundaries?
  - Already apparent from these examples: The output does not have the same size as the image by default
  - Why? Images have finite "support" on our computers
  - Support of image: $\{(m,n) \in \mathbb{Z}^2 : f[m,n] \neq 0\}$
  - With more edge processing

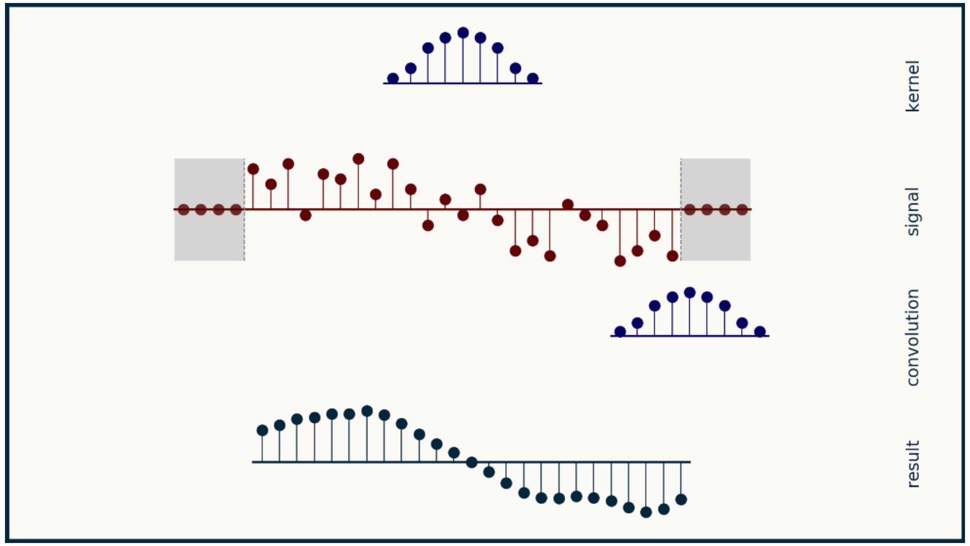


Convolution Sum

- What happens in practice at these boundaries?
  - Nothing is an option → gives the above result (usually not desirable)
  - Zero-padding



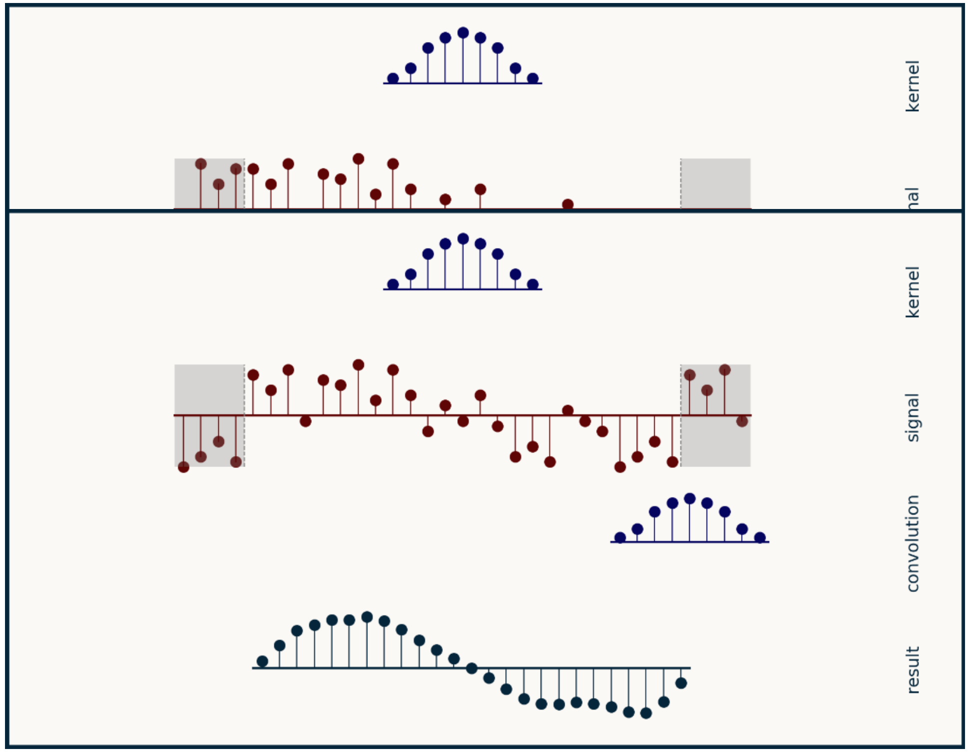




https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1 https://e2eml.school/convolution_one_d.html

Convolution Sum

- What happens in practice at these boundaries?
  - Nothing is an option → gives the above result (usually not desirable)
  - Zero-padding
  - Mirror extension
    - 2D discrete cosine transform
  - Circular extension
    - 2D discrete Fourier transform



https://e2eml.school/convolution_one_d.html

Convolution Sum

- Properties

$$f[m,n] * h[m,n] = h[m,n] * f[m,n]$$ commutative

$$(f[m,n] * g[m,n]) * h[m,n] = f[m,n] * (g[m,n] * h[m,n])$$ associative

$$f[m,n] * (h_1[m,n] + h_2[m,n]) = f[m,n] * h_1[m,n] + f[m,n] * h_2[m,n]$$ distributive

$$(f_1[m]f_2[n]) * (h_1[m]h_2[n]) = (f_1[m] *_{1D} h_1[m])(f_2[n] *_{1D} h_2[n])$$ separability

2D discrete-space Fourier Transform

- This is again motivated as eigenfunctions of LSI systems
  - Analogous to continuous-space
  - So we will skip to the derivation

$$F(\omega_x, \omega_y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f[m,n] e^{-i(\omega_x m + \omega_y n)}$$

  - Inverse is defined via an integral (we will skip this too)

- Properties
  - Periodic: $F(\omega_x, \omega_y) = F(\omega_x + 2\pi, \omega_y) = F(\omega_x, \omega_y + 2\pi)$
  - Transpose

$$f[m,n] \xleftrightarrow{\mathcal{F}} F(\omega_x, \omega_y)$$

$$f[n,m] \xleftrightarrow{\mathcal{F}} F(\omega_y, \omega_x)$$

2D discrete-space Fourier Transform

- Multiplication

$$f[m,n]g[m,n] \xleftrightarrow{\mathcal{F}} \frac{1}{(2\pi)^2}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi} F(\lambda_x,\lambda_y)G(\omega_x-\lambda_x,\omega_y-\lambda_y)d\lambda_x d\lambda_y$$

2π-periodic convolution in Fourier domain

- Convolution

$$f[m,n]*g[m,n] \xleftrightarrow{\mathcal{F}} F(\omega_x,\omega_y)G(\omega_x,\omega_y)$$

- Magnitude and phase
  - In general $F(\omega_x,\omega_y)$ is complex, i.e. $F(\omega_x,\omega_y)=F_R(\omega_x,\omega_y)+iF_I(\omega_x,\omega_y)$
  - As with other complex numbers, we can write this: $F(\omega_x,\omega_y)=|F(\omega_x,\omega_y)|e^{i\theta(\omega_x,\omega_y)}$
  - Normally both available
  - Some applications: One may be available (e.g. phase retrieval)

LSI Systems in Frequency Domain

- We have seen Fourier analysis over the past two lectures
  - So far, it helped us understand the frequency properties of images
  - But we can also use it for “system” or “filter” design
  - These will be simple pre-processing tools, e.g.
    - Noise reduction
      - Low-pass filter (why?)
    - Edge enhancement
      - High-pass filter (why?)
  - This understanding/intuition will come in handy when developing more complex image processing systems

LSI Systems in Frequency Domain

- Example (1D):

$$h[n] = \begin{bmatrix} \underline{\frac{1}{4}} & \frac{1}{2} & \frac{1}{4} \end{bmatrix}$$

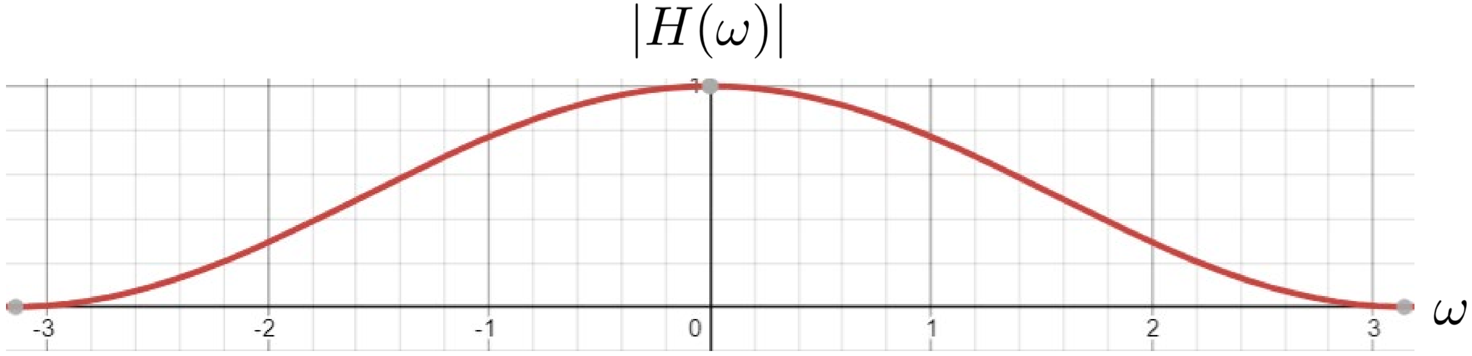
underline denotes 0th location

$$H(\omega) = \sum_n h[n]e^{-i\omega n} = \frac{1}{4} + \frac{1}{2}e^{-i\omega} + \frac{1}{4}e^{-i2\omega}$$

$$= \frac{1}{4}e^{-i\omega}\left(e^{i\omega} + 2 + e^{-i\omega}\right) = \frac{1}{2}e^{-i\omega}(1 + \cos\omega)$$

$$|H(\omega)| = \frac{1}{2}(1 + \cos\omega)$$

$$\angle H(\omega) = -\omega$$

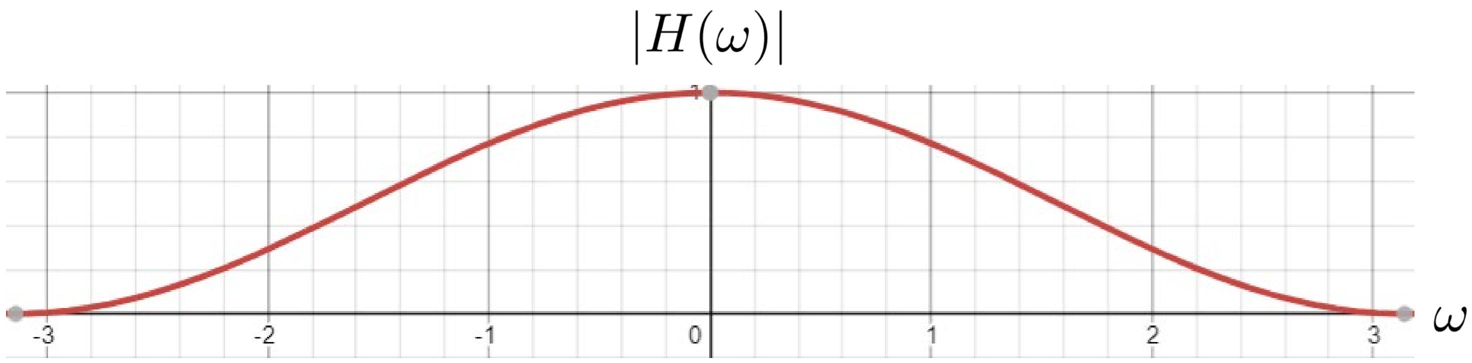


LSI Systems in Frequency Domain

- Example (1D):

$$h[n] = \begin{bmatrix} \underline{\frac{1}{4}} & \frac{1}{2} & \frac{1}{4} \end{bmatrix}$$

$$|H(\omega)| = \frac{1}{2}(1 + \cos\omega)$$
$$\angle H(\omega) = -\omega$$



  - What do these frequencies mean?
    - Recall that DSFT is 2π periodic
    - 0 is still the lowest frequency
    - ±π are the high frequencies
    - This filter is 0 at ±π → Low-pass filter

LSI Systems in Frequency Domain

- Example (Now in 2D):

$$h[m,n] = \begin{bmatrix} 0 & \frac{1}{4} & 0 \\ \frac{1}{4} & \underline{\frac{1}{2}} & \frac{1}{4} \\ 0 & \frac{1}{4} & 0 \end{bmatrix}$$

$$\begin{aligned} H(\omega_x, \omega_y) &= \frac{1}{2} + \frac{1}{4}\left(e^{-i\omega_x} + e^{i\omega_x} + e^{-i\omega_y} + e^{i\omega_y}\right) \\ &= \frac{1}{2}(1 + \cos\omega_x + \cos\omega_y) \end{aligned}$$

  - Check high-frequency performance

$$|H(\pi, 0)| = \frac{1}{2} \neq 0$$

    - Not *ideal* low-pass

LSI Systems in Frequency Domain

- Example (Moving average):

$$h[m,n] = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & \underline{1} & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$H(\omega_x, \omega_y) = \frac{1}{9}(1 + 2\cos\omega_x)(1 + 2\cos\omega_y)$$

  - Check high-frequency performance

$$|H(\pi, 0)| = \frac{1}{3} \neq 0$$

    - Also not *ideal* low-pass

LSI Systems in Frequency Domain

- Example (Separable):
  - We already saw a better 1D low-pass filter (0 at ±π)
  - We can build a 2D filter from it

$$h[m,n] = \begin{bmatrix} \frac{1}{4} \\ \frac{1}{2} \\ \frac{1}{4} \end{bmatrix} \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix} = \begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix}$$

$$H(\omega_x, \omega_y) = \frac{1}{4}(1 + \cos\omega_x)(1 + \cos\omega_y)$$

  - Check performance

$$H(0,0) = 1$$
$$H(\pm\pi, \omega_y) = 0 = H(\omega_x, \pm\pi)$$

Yet Another Fourier Transform

- DSFT is useful for simple system analysis
- In reality: All our images have finite support (extent)
  - We do not need to sum from $\pm\infty$
- We need one final tool (which is what we will actually use)
  - Discrete Fourier transform (DFT)

Discrete-space Fourier Series

- To get to DFT, first consider a periodic discrete-space signal

$$\tilde{x}[m,n] \text{ periodic with period } (M,N)$$

- We follow the same process as in continuous-space, defining

$$\phi_{k,l}[m,n] = e^{i2\pi\left(k\frac{m}{M}+l\frac{n}{N}\right)}$$

- Do the projection argument…

$$\tilde{x}[m,n] = \sum_{k,l} c_{k,l}\phi_{k,l}[m,n]$$

$$\text{Note } e^{-i2\pi(k+M)\frac{m}{M}} = e^{-i2\pi k\frac{m}{M}}\underbrace{e^{-i2\pi M\frac{m}{M}}}_{e^{-i2\pi m}}$$

=1 since *m* is an integer

- So we only need the summation w.r.t. (*k*,*l*) over one period!

Discrete-space Fourier Series

- We will not do the whole derivation, but this leads to the following:

$\tilde{x}[m,n]$ periodic with period (M,N)

Define $\quad \tilde{X}[k,l] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \tilde{x}[m,n]\, e^{-i2\pi\left(k\frac{m}{M}+l\frac{n}{N}\right)}$

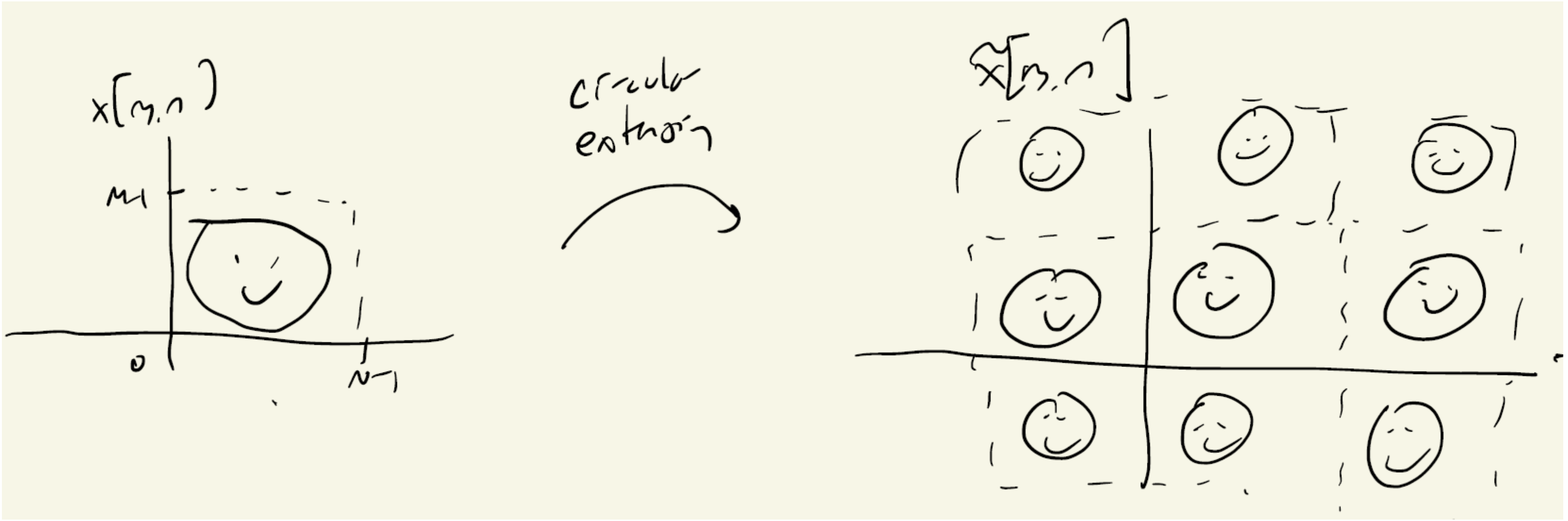
Note $e^{-i2\pi(k+M)\frac{m}{M}} = e^{-i2\pi k\frac{m}{M}} \underbrace{e^{-i2\pi M\frac{m}{M}}}_{e^{-i2\pi m}}$ ⟵ =1 since *m* is an integer

Thus $\tilde{X}[k,l]$ itself is periodic with period (M,N)

and $\quad \tilde{x}[m,n] = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} \tilde{X}[k,l]\, e^{i2\pi\left(k\frac{m}{M}+l\frac{n}{N}\right)}$

Discrete Fourier Transform

- What if x[m,n] is not periodic, but finite *M*×*N* support (as most our images do)?
  - Make it periodic
  - (*M*,*N*)-point circular extension



Discrete Fourier Transform

- Circular extension

$$\tilde{x}[m,n] \triangleq x[m \mod M, n \mod N]$$

depends on $x[m,n]$ for

$m \in \{0, \ldots, M-1\}$

$n \in \{0, \ldots, M-1\}$

- How to get the image back?

$$x[m,n] = \tilde{x}[m,n] R_{M\times N}[m,n]$$

rectangular window in image domain

Discrete Fourier Transform

- Take the DSFT of the circular extension

$$\tilde{X}[k,l] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \tilde{x}\,[m,n]\, e^{-i2\pi\left(k\frac{m}{M}+l\frac{n}{N}\right)}$$

- DFT is when this is truncated

$$X[k,l] = \tilde{X}[k,l] R_{M\times N}[k,l]$$

- Everything we need to store is only defined over *M*×*N* regions

$$X[k,l] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x\,[m,n]\, e^{-i2\pi\left(k\frac{m}{M}+l\frac{n}{N}\right)} \qquad k \in \{0,\ldots,M-1\}, l \in \{0,\ldots,N-1\}$$

$$x[m,n] = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{k=0}^{N-1} X\,[k,l]\, e^{i2\pi\left(k\frac{m}{M}+l\frac{n}{N}\right)} \qquad m \in \{0,\ldots,M-1\}, n \in \{0,\ldots,N-1\}$$

Discrete Fourier Transform

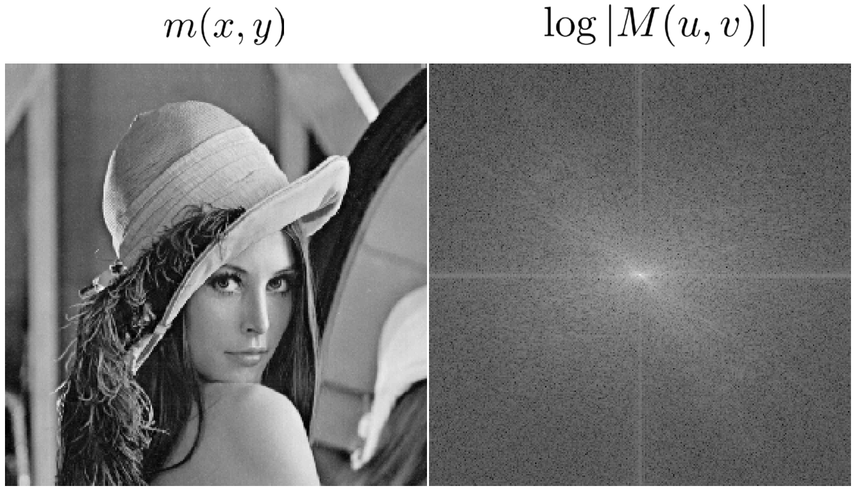
- Multiplication in DFT domain → *Circular* convolution in image space

$$X[k,l]H[k,l] \xleftrightarrow{DFT} x[m,n] \circledast h[m,n]$$
$$= \sum_{m'=0}^{M-1} \sum_{n'=0}^{N-1} x[m',n']h[(m-m') \mod M, (n-n') \mod N]$$

- i.e. wrap-around
- Different than linear convolution (boundary conditions)
- If one wants to evaluate linear convolution (without wrap-around) → need zero-padding (see earlier)
- Note unlike what you heard in EE 3015, in most image processing applications, it is faster to do convolutions in image space
  - The convolution filters usually have very small support (e.g. 3×3)

Discrete Fourier Transform

- Why DFT?
  - Allows us to analyze the spatial frequency spectrum of images
  - e.g. from last lecture: All generated with DFT



  - DFT can be implemented very fast
  - Using fast Fourier transform (FFT)

Brute force evaluation of DFT: $O((MN)^2)$ operations
FFT allows $O((MN)\log(\max(M,N)))$ operations

from EE 4541

  - FFT is what we use in practice

Fourier Magnitude and Phase

f(x,y) angle(F(u,v))

g(x,y) log|G(u,v)|

$\text{ifft2}(|G(u,v)|e^{i\cdot angle(F(u,v))})$

Discrete Fourier Transform

- It will also be desirable to look at DFT in matrix-vector notation
  - First let's do the 1D case $$X[k] = \sum_{m=0}^{M-1} x\,[m]\, e^{-i2\pi\left(k\,\frac{m}{M}\right)}$$
  - Define matrix $\mathbf{W}_M$ with entries $W_{k,n} = \left(e^{-i\frac{2\pi}{M}}\right)^{kn}$
  - Then

$$\begin{bmatrix} X[0] \\ X[1] \\ \vdots \\ X[N-1] \end{bmatrix} = \begin{bmatrix} & & \\ & \mathbf{W}_M & \\ & & \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}$$

  - By properties of $\mathbf{W}_M$, we have

$$\mathbf{W}_M^H \mathbf{W}_M = M\,\mathbf{I}_M = \mathbf{W}_M \mathbf{W}_M^H \qquad \Rightarrow \mathbf{W}_M^{-1} = \frac{1}{M}\mathbf{W}_M^H$$

conjugate transpose identity matrix

Discrete Fourier Transform

- Similarly easy to deduce a Parseval relation in this form

$$
\begin{aligned}
\mathbf{x}^H \mathbf{y} &= \left(\mathbf{W}_M^{-1}\mathbf{X}\right)^H \left(\mathbf{W}_M^{-1}\mathbf{Y}\right) \\
&= \left(\frac{1}{M}\mathbf{W}_M^H \mathbf{X}\right)^H \left(\frac{1}{M}\mathbf{W}_M^H \mathbf{Y}\right) \\
&= \frac{1}{M^2}\mathbf{X}^H \underbrace{\mathbf{W}_M \mathbf{W}_M^H}_{M\mathbf{I}_M} \mathbf{Y} = \frac{1}{M}\mathbf{X}^H \mathbf{Y}
\end{aligned}
$$

Discrete Fourier Transform

- It will also be desirable to look at DFT in matrix-vector notation
  - In 2D

$$\mathbf{x}_{\text{image}} = \begin{bmatrix} x[0,0] & \cdots & x[0,N-1] \\ \vdots & & \\ x[M-1,0] & \cdots & x[M-1,N-1] \end{bmatrix}$$

  - By separability of the DFT, we have

$$\mathbf{X}_{\text{image}} = \mathbf{W}_M \mathbf{x}_{\text{image}} \mathbf{W}_N^T$$

Discrete Fourier Transform

- But often, it will be more convenient for us to *vectorize* the image

$$\mathbf{x}_{\text{vec}} = \text{vec}(\mathbf{x}_{\text{image}}) = \begin{bmatrix} x[0,0] \\ \vdots \\ x[M-1,0] \\ x[0,1] \\ \vdots \\ x[M-1,N-1] \end{bmatrix}$$

- In that case

$$\mathbf{X}_{\text{vec}} = \left(\mathbf{W}_M \otimes \mathbf{W}_N\right)\mathbf{x}_{\text{vec}}$$

Note this is a linear operator
i.e. $\mathbf{X}_{\text{vec}} = \mathbf{L}\mathbf{x}_{\text{vec}}$
for linear operator
$\mathbf{L} = \mathbf{W}_M \otimes \mathbf{W}_N$

where

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1N}\mathbf{B} \\ \vdots & & \\ a_{m1}\mathbf{B} & \cdots & a_{mN}\mathbf{B} \end{bmatrix}$$

is the Kronecker product

Discrete Fourier Transform

- What happens to noise in the Fourier domain?
  - Some basic probability
    - Consider $\mathbf{y} = \mathbf{A}\mathbf{x}$

      random vector — $\mathbf{y}$; matrix — $\mathbf{A}$; random vector — $\mathbf{x}$

    - Recall covariance matrix of **y** is given as $\mathrm{Cov}\{\mathbf{y}\} = \mathrm{Cov}\{\mathbf{A}\mathbf{x}\} = \mathbf{A}\mathrm{Cov}\{\mathbf{x}\}\mathbf{A}^H$
    - Suppose random vector **x** has unorrelated entries with same variance, i.e. $\mathrm{Cov}\{\mathbf{x}\} = \sigma^2\mathbf{I}_N$
    - Then its DFT $\mathbf{X} = \mathbf{W}_N\mathbf{x}$ has covariance matrix

$$\mathrm{Cov}\{\mathbf{X}\} = \mathbf{W}_N\mathrm{Cov}\{\mathbf{x}\}\mathbf{W}_N^H = \mathbf{W}_N\sigma^2\mathbf{I}_N\mathbf{W}_N^H = \sigma^2\mathbf{W}_N\mathbf{W}_N^H = \sigma^2 \cdot N \cdot \mathbf{I}_N$$

    - Therefore uncorrelated in image domain ↔ uncorrelated in Fourier domain
  - Note: This applies to any orthogonal/unitary transformation

Recap of 2D Signals & Systems

- Concept of signals & systems
  - Images are 2D signals
  - Image formation or processing systems
- Fourier analysis review
  - Continuous-space (Fourier series, Fourier transform)
  - Sampling & interpolation
  - Discrete-space (DSFT, DFT/FFT)
  - How Fourier analysis (via DFT) describes images
    - Low-frequency content → contrast information, most of the energy of the image
    - High-frequency content → edges and other sharper structures
- These intuitions & Fourier analysis will be useful for algorithm design
- Next: Start on image enhancement