Lecture 4
Signal Processing
6.003 Signals and systems

Time continuous signal

Time discrete signal

Review class notes!
A 2D discrete signal

\[ I = \begin{bmatrix}
149 & 164 & 172 & 175 & 178 & 179 & 176 & 118 & 97 & 168 & 175 & 171 & 169 & 175 & 176 & 177 & 165 & 152 \\
173 & 164 & 158 & 165 & 180 & 180 & 150 & 89 & 61 & 34 & 137 & 186 & 186 & 182 & 175 & 165 & 160 & 164 \\
135 & 132 & 131 & 125 & 115 & 129 & 132 & 74 & 54 & 41 & 104 & 156 & 152 & 156 & 164 & 156 & 141 & 144 \\
\end{bmatrix} \]
A 2D discrete signal
A 2D discrete signal

A tiny person of 18 x 18 pixels
Images are turned into column vectors by concatenating all image columns.
Signal / image space

Scalar product between two signals $f$, $g$:

$$\langle f, g \rangle = \sum_{n=0}^{N-1} f[n] g^*[n] = f^T g^*$$

L2 norm of $f$:

$$E_f = \|f\|^2 = \langle f, f \rangle = \sum_{n=0}^{N-1} |f[n]|^2 = f^T f^*$$

Distance between two signals $f$, $g$:

$$d_{f,g}^2 = \|f - g\|^2 = \sum_{n=0}^{N-1} |f[n] - g[n]|^2 = E_f + E_g - 2 \langle f, g \rangle$$
Filtering

We want to remove unwanted sources of variation, and keep the information relevant for whatever task we need to solve.
For a filter, $H$, to be linear, it has to verify:

\[ H(a[m,n] + b[m,n]) = H(a[m,n]) + H(b[m,n]) \]

\[ H(Ca[m,n]) = C \cdot H(a[m,n]) \]
Linear filtering

A linear filter in its most general form can be written as (for a 1D signal of length $N$):

$$f[n] = \sum_{k=0}^{N-1} h[n, k] g[k]$$

It is useful to write this as a matrix and vector multiplications:

$$\begin{bmatrix} f[0] \\ f[1] \\ \vdots \\ f[M-1] \end{bmatrix} = \begin{bmatrix} h[0,0] & h[0,1] & \cdots & h[0,N-1] \\ h[1,0] & h[1,1] & \cdots & h[1,N-1] \\ \vdots & \vdots & \ddots & \vdots \\ h[M-1,0] & h[M-1,1] & \cdots & h[M-1,N-1] \end{bmatrix} \begin{bmatrix} g[0] \\ g[1] \\ \vdots \\ g[N-1] \end{bmatrix}$$
A linear filter in its most general form can be written as (for a 2D image of size $N \times M$):

$$f[n,m] = \sum_{k,l=0}^{N-1,M-1} h[n,m,k,l] g[k,l]$$

Which can also be written in matrix form as in the 1D case:
We need translation invariance
A translation invariant filter

Example: The output for the sample $n$ is twice the value of the input at that same time minus the sum of the two previous time steps.

\[
\begin{align*}
    f[0] &= 2g[0] - g[-1] - g[-2] \\
    f[1] &= 2g[1] - g[0] - g[-1] \\
    \vdots \\
    f[n] &= 2g[n] - g[n-1] - g[n-2]
\end{align*}
\]

A filter is linear translation invariant (LTI) if it is linear and when we translate the input signal by $m$ samples, the output is also translated by $m$ samples.
Convolution

\[ f[n] = h \circ g = \sum_{k=0}^{N-1} h[n-k] g[k] \]

For the previous example: \( h = [2, -1, -1] \)
Properties of the convolution

Commutative

\[ h[n] \circ g[n] = g[n] \circ h[n] \]

Associative

\[ h[n] \circ g[n] \circ q[n] = h[n] \circ (g[n] \circ q[n]) = (h[n] \circ g[n]) \circ q[n] \]

Distributive with respect to the sum

\[ h[n] \circ (f[n] + g[n]) = h[n] \circ f[n] + h[n] \circ g[n] \]

Shift property

\[ f[n - n_0] = h[n] \circ g[n - n_0] = h[n - n_0] \circ g[n] \]
A translation invariant filter

The same weighting occurs within each window
2D convolution

Convolution weights

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]
2D convolution

Input image

Convolution output
2D convolution

Input image

Convolution output

\[
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
\end{array}
\]
2D convolution

Input image

Convolution output
2D convolution

Input image

Convolution output

\[ g = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \]
2D convolution

Input image

Convolution output
2D convolution

Input image

Convolution output
What does it do?
• Replaces each pixel with an average of its neighborhood
• Achieve smoothing effect (remove sharp features)
In the 1D case, it helps to make explicit the structure of the matrix:

\[ 0 \quad 0 \quad 0 \quad 90 \quad 90 \quad 90 \quad 90 \quad 0 \quad 0 \]

\[ \begin{bmatrix} 1/3 & 1/3 & 1/3 \end{bmatrix} \]

\[ = \]

\[ 0 \quad 20 \quad 40 \quad 60 \quad 60 \quad 60 \quad 40 \quad 20 \]
Convolution

In the 1D case, it helps to make explicit the structure of the matrix:

\[
\begin{bmatrix}
0 & 0 & 0 & 90 & 90 & 90 & 90 & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
0 & 20 & 40 & 60 & 60 & 60 & 40 & 20 \\
\end{bmatrix}
= 
\begin{bmatrix}
\frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\
\end{bmatrix}
\]
Convolution

In the 1D case, it helps to make explicit the structure of the matrix:

\[
\begin{bmatrix}
0 & 0 & 0 & 90 & 90 & 90 & 90 & 0 & 0
\end{bmatrix}
\circ
\begin{bmatrix}
1/3 & 1/3 & 1/3
\end{bmatrix}
= \begin{bmatrix}
0 & 20 & 40 & 60 & 60 & 60 & 40 & 20
\end{bmatrix}
\]

In the 1D case, it helps to make explicit the structure of the matrix:

\[
\begin{bmatrix}
f[1] \\
f[2] \\
\vdots \\
f[8]
\end{bmatrix}
\circ
\begin{bmatrix}
1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
& & & & & & & & & \\
& & & & & & & & & \\
& & & & & & & & & \\
& & & & & & & & & \\
& & & & & & & & & \\
& & & & & & & & & \\
& & & & & & & & & \\
& & & & & & & & & \\
& & & & & & & & & \\
\end{bmatrix}
= \begin{bmatrix}
g[0] \\
g[1] \\
\vdots \\
g[9]
\end{bmatrix}
\]
In the 1D case, it helps to make explicit the structure of the matrix:

\[
\begin{bmatrix}
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\frac{1}{3} & \frac{1}{3} & \frac{1}{3}
\end{bmatrix}
= 
\begin{bmatrix}
0 & 20 & 40 & 60 & 60 & 60 & 40 & 20
\end{bmatrix}
\]
Convolutional Neural Networks

- Neural network with specialized connectivity structure

LeCun et al. 1989
Rectangular filter

\[ g[m,n] \times h[m,n] = f[m,n] \]
Rectangular filter
Rectangular filter

\[ g[m,n] \times h[m,n] = f[m,n] \]
Important signals

The impulse

The result of convolving a signal $g[n]$ with the impulse signal is the same signal:

$$f[n] = \delta \circ g = \sum_k \delta[n-k]g[k] = g[n]$$

Convolving a signal $f$ with a translated impulse $\delta[n-n_0]$ results in a translated signal:

$$f[n-n_0] = \delta[n-n_0] \circ f[n]$$
Why the impulse is so important

\[ f[n] = \sum_{k} f[k] \delta[n - k] \] Write the input signal as a sum of impulses

\[ g[n] = \sum_{k} f[k] h[n - k] = f \ O h = h \ O f \] Then the output of an LTI system is the corresponding sum of impulse responses
Why the impulse is so important
Why the impulse is so important
Examples

The impulse signal

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Examples (using zero padding)
Examples
Handling boundaries
Handling boundaries

Zero padding

11x11 ones
Handling boundaries

### Input
- Zero padding
- Circular repetition
- Mirror edge pixels
- Repeat edge pixels
- Ground truth

### Output

### Error
Image transformations

1. From pixels to edges
2. From edges to geometric primitives

- Edge normals
- 3D orientation
- Contact edges
- Depth discontinuities

Input image

Edge strength

X

Y

Z
Linear image transformations

In analyzing images, it’s often useful to make a change of basis.

\[ \tilde{F} = U\tilde{f} \]

Transformed image → Fourier transform, or Wavelet transform, or Steerable pyramid transform

Vectorized image

\[ U \]

=
The Discrete Fourier transform

Discrete Fourier Transform (DFT) transforms a signal \( f[n] \) into \( F[u] \) as:

\[
F[u] = \sum_{n=0}^{N-1} f[n] \exp \left( -2\pi j \frac{u n}{N} \right)
\]

The inverse of the DFT is:

\[
f[n] = \frac{1}{N} \sum_{u=0}^{N-1} F[u] \exp \left( 2\pi j \frac{u n}{N} \right)
\]

The signal \( f[n] \) is a weighted linear combination of complex exponentials with weights \( F[u] \).
The Discrete Fourier transform

Discrete Fourier Transform (DFT) transforms a signal $f[n]$ into $F[u]$ as:

$$F[u] = \sum_{n=0}^{N-1} f[n] \exp\left(-2\pi j \frac{un}{N}\right)$$

Discrete Fourier Transform (DFT) is a linear operator. Therefore, we can write:

$\text{Lets visualize the transform coefficients}$

$$F = \begin{bmatrix}
\exp\left(-2\pi j \frac{un}{N}\right) & \ldots & \\
\end{bmatrix}$$

$$f$$

$\text{NxN array}$
Visualizing the Fourier transform

\[ F[u] = \sum_{n=0}^{N-1} f[n] \exp\left(-2\pi j \frac{un}{N}\right) \]

\[ \exp(\alpha j) = \cos(\alpha) + j \sin(\alpha) \]

\[ \cos\left(2\pi \frac{un}{N}\right) - j \sin\left(2\pi \frac{un}{N}\right) \]

For:
\[ u=1 \]
\[ N=16 \]
Visualizing the transform coefficients

\[ \exp \left( -2\pi j \frac{un}{N} \right) \quad \text{For } N=16 \]
Visualizing the transform coefficients

\[ \exp\left( -2\pi j \frac{un}{N} \right) \quad \text{For } N=16 \]
Visualizing the transform coefficients

\[ \exp\left(-2\pi j \frac{un}{N}\right) \]  For \( N=16 \)

\[ F_{f_{16x16}} = \begin{bmatrix}
    n=0 & 1 & 2 & \cdots & n=15 \\
    u=0 & & & & \\
    u=1 & & & & \\
    u=2 & & & & \\
    u=15 & & & &
\end{bmatrix} \]
Visualizing the transform coefficients

\[ \exp \left( -2\pi j \frac{un}{N} \right) \quad \text{For } N=16 \]

For a 16x16 array, the transform coefficients are arranged as shown in the diagram.
Visualizing the transform coefficients

$$\exp\left(-2\pi j \frac{un}{N}\right)$$ For $N=16$

![Diagram showing visualizing the transform coefficients](image)
Visualizing the transform coefficients

$$\exp\left(-2\pi j \frac{un}{N}\right)$$ For $N=16$

![Diagram showing a 16x16 array with real and imaginary components labeled from 0 to 15, with a color scale representing the transform coefficients. The array is structured with rows and columns, and the colors vary across the array, indicating the magnitude of the coefficients.]
Visualizing the transform coefficients

$$\exp\left(-2\pi j \frac{un}{N}\right)$$

For $N=16$

![Diagram showing the visualization of transform coefficients with a 16x16 array and a color spectrum ranging from red to blue, with labels for both real and imaginary parts.]
Visualizing the transform coefficients

\[ \exp \left( -2\pi j \frac{un}{N} \right) \]

For \( N=16 \)

16x16 array

Real

Imag

F =

16x16 array
The inverse of the Discrete Fourier transform

Discrete Fourier Transform (DFT):

\[
F[u] = \sum_{n=0}^{N-1} f[n] \exp\left(-2\pi j \frac{un}{N}\right)
\]

Its inverse:

\[
f[n] = \frac{1}{N} \sum_{u=0}^{N-1} F[u] \exp\left(2\pi j \frac{un}{N}\right)
\]
For images, the 2D DFT

1D Discrete Fourier Transform (DFT) transforms a signal $f[n]$ into $F[u]$ as:

$$F[u] = \sum_{n=0}^{N-1} f[n] \exp\left(-2\pi j \frac{un}{N}\right)$$

2D Discrete Fourier Transform (DFT) transforms an image $f[n,m]$ into $F[u,v]$ as:

$$F[u, v] = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[n, m] \exp\left(-2\pi j \left(\frac{un}{N} + \frac{vm}{M}\right)\right)$$
Visualizing the 2D DFT coefficients

\[ F[u, v] = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[n, m] \exp\left(-2\pi j \left( \frac{un}{N} + \frac{vm}{M} \right) \right) \]

For \( N=M=16 \)

256x256 array
A remarkable property of Fourier transform

In the spatial domain, the output of \( f \) is the convolution:

\[
f [m, n] = h \circ g = \sum_{k,l} h [m - k, n - l] g [k, l]
\]

In the frequency domain:

\[
\]

Terminology:

Impulse response: \( h [m,n] \)
Transfer function: \( H [u, v] \)
**Dual convolution property**

The Fourier transform of the convolution is the product of Fourier transforms

\[ f[m, n] = h \circ g \quad \iff \quad F[u, v] = G[u, v] H[u, v] \]

The Fourier transform of the product is the convolution of Fourier transforms

\[ f[n, m] = g[n, m] h[n, m] \quad \iff \quad F[u, v] = \frac{1}{NM} G[u, v] \circ H[u, v] \]
Visualizing the image Fourier transform

\[
F[u, v] = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[n, m] \exp \left( -2\pi j \left( \frac{un}{N} + \frac{vm}{M} \right) \right)
\]

The values of \( F[u,v] \) are complex.

Using the real and imaginary components:

\[
\]

Or using a polar decomposition:

\[
F[u, v] = A[u, v] \exp (j \theta[u, v])
\]

Amplitude \hspace{2cm} Phase
Visualizing the image Fourier transform

\[ f[n,m] \]

\[ F[u,v] \]
Simple Fourier transforms

Image

DFT (amplitude)

\[ \cos \left( 2\pi \left( \frac{u_0 n}{N} + \frac{v_0 m}{M} \right) \right) \quad \longleftrightarrow \quad \frac{1}{2} \left( \delta [u - u_0, v - v_0] + \delta [u + u_0, v + v_0] \right) \]
Images are 64x64 pixels. The wave is a cosine, therefore DFT phase is zero.
Some important Fourier transforms

Image

Magnitude DFT

Phase DFT

64x64 pixels
Some important Fourier transforms

Image

Magnitude DFT

Phase DFT

64x64 pixels
Some important Fourier transforms

Image

Magnitude DFT

Phase DFT

64x64 pixels
Some important Fourier transforms

Image

Magnitude DFT

Phase DFT

64x64 pixels

\[ \delta_k [n, m] = \sum_{s=0}^{N/k-1} \sum_{r=0}^{M/k-1} \delta [n - sk, m - rk] \]

delta train, delta comb, impulse train

\[ \Delta_k [u, v] = \frac{NM}{k^2} \sum_{s=0}^{k-1} \sum_{r=0}^{k-1} \delta \left[ u - s \frac{N}{k}, v - r \frac{M}{k} \right] \]

when \( M \) and \( N \) are divisible by \( k \)
Some important Fourier transforms

Image

Magnitude DFT

Phase DFT

Translation

Shifts of an image only produce changes on the phase of the DFT.
Some important Fourier transforms

Image

Magnitude DFT

Phase DFT

Scale

Small image details produce content in high spatial frequencies
Some important Fourier transforms

Orientation

A line transforms to a line oriented perpendicularly to the first.
The Fourier Transform of some important images

Image

Log(1+Magnitude FT)
The DFT Game: find the right pairs

Images

DFT magnitude

fx(cycles/image pixel size)

fx(cycles/image pixel size)

fx(cycles/image pixel size)
The DFT Game: find the right pairs

(Solution in the class notes)
The inverse Discrete Fourier transform

2D Discrete Fourier Transform (DFT) transforms an image $f[n,m]$ into $F[u,v]$ as:

$$F[u,v] = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[n,m] \exp \left( -2\pi j \left( \frac{un}{N} + \frac{vm}{M} \right) \right)$$

The inverse of the 2D DFT is:

$$f[n,m] = \frac{1}{NM} \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} F[u,v] \exp \left( +2\pi j \left( \frac{un}{N} + \frac{vm}{M} \right) \right)$$

How does summing waves ends up giving back a picture?
#1: Range [0, 1]
Dimensions [256, 256]

#2: Range [0.000108, 0.0257]
Dimensions [256, 256]
6

#1: Range [0, 1]
Dims [256, 256]

#2: Range [1.899 0.07, 0.226]
Dims [256, 256]
\( \Psi_1: \text{Range } [0, 1] \\
\text{Dims } [268, 256] \\
\)

\( \Psi_2: \text{Range } [4.79e-007, 0.633] \\
\text{Dims } [256, 256] \\
\)
538

538

#1: Range [0, 1]
Dims [256, 256]

#2: Range [0.17e-036, 8.4]
Dims [256, 256]
2094

#1: Range [0, 1]
Dims [256, 256]

#2: Range [8.7e-05, 19]
Dims [256, 256]
4052.
8056.
28743

#1: Range [0, 1]
Dims [256, 256]

#2: Range [0.001, 146]
Dims [256, 256]
65536.
Now, an analogous sequence of images, but selecting Fourier components in descending order of magnitude.
Figure 13

#1: Range [0, 1]
Dims [256, 256]

#2: Range [1.76e-005, 1.26]
Dims [256, 256]
Figure 14

#1: Range [0, 1]
Dims [256, 256]

#2: Range [2.24e-005, 1.28]
Dims [256, 256]
Figure 15

#1: Range [0, 1]
Dims [256, 256]

#2: Range [0.000347, 1.27]
Dims [256, 256]
#1: Range [0, 1]
Dims [256, 256]

#2: Range [0.000522, 1.23]
Dims [256, 256]
Figure 18

#1: Range [0, 1]
Dims [256, 256]

#2: Range [0.000385, 1.1]
Dims [256, 256]

16385
Figure 19

32769

#1. Range [0, 1]
Dims [256, 256]

#2. Range [0.0246, 1.03]
Dims [256, 256]
Figure 20

#1. Range [0.5, 1.5]
Dims [256, 256]

#2. Range [0.029, 1]
Dims [256, 256]
Visualizing the image Fourier transform

$f[n,m]$

$F[u,v]$
Phase and Magnitude

\[ F[u, v] = A[u, v] \exp(j \theta[u, v]) \]

Each color channel is processed in the same way.
Phase and Magnitude

• Curious fact
  – all natural images have about the same magnitude transform
  – hence, phase seems to matter, but magnitude largely doesn’t
Some visual areas...

From M. Lewicky
Figure 1. Stimulus presentation scheme. The stimuli were originally calibrated to be seen at a distance of 150 cm in a 19" display.
Let’s define the following image:

\[ I[n,m] = A[n] \sin(2\pi f[m] m/M) \]

With:

\[ A[n] = A_{\text{min}} \left( \frac{A_{\text{max}}}{A_{\text{min}}} \right)^{n/N} \]

\[ f[m] = f_{\text{min}} \left( \frac{f_{\text{max}}}{f_{\text{min}}} \right)^{m/M} \]

What do you think you should see when looking at this image?
\[ I[n, m] = A[n] \sin(2\pi f[m] m/M) \]
\[ I[n,m] = A[n] \sin(2\pi f[m] m/M) \]
Contrast Sensitivity Function

Blackmore & Campbell (1969)

Maximum sensitivity

~ 6 cycles / degree of visual angle

Invisible

visible

Contrast sensitivity

Spatial frequency (cycles/degree)

Low

High

0.1

1

10

100

Things that are very close and/or large are hard to see

Things far away are hard to see
Vasarely visual illusion