Frequency Representations of Images

- 2D Fourier Representations
- Convolution in 2D
Quiz 2

Thursday, 21 November, 7:30-9:30pm, room TBD

No recitation next week, no lecture on Thursday 21 Nov.

The quiz is closed book.
No electronic devices (including calculators).

You may use two 8.5x11” sheets of notes (handwritten or printed, front and back).

Coverage: lectures, recitations, homeworks, and labs up to and including 20 Nov.

Practice materials (old exams) are available from the web site.
Review: Fourier Representations

One dimensional DTFT:

\[ F(\Omega) = \sum_{n=-\infty}^{\infty} f[n] e^{-j\Omega n} \]

\[ f[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} F(\Omega) e^{j\Omega n} d\Omega \]

Two dimensional DTFT:

\[ F(\Omega_r, \Omega_c) = \sum_{r=-\infty}^{\infty} \sum_{c=-\infty}^{\infty} f[r, c] e^{-j(\Omega_r r + \Omega_c c)} \]

\[ f[r, c] = \frac{1}{4\pi^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} F(\Omega_r, \Omega_c) e^{j(\Omega_r r + \Omega_c c)} d\Omega_r d\Omega_c \]

\( r \) and \( c \) are discrete spatial variables (units: pixels)
\( \Omega_r \) and \( \Omega_c \) are spatial frequencies (units: radians / pixel)
Review: Fourier Representations

One dimensional DFT:

\[ F[k] = \frac{1}{N} \sum_{n=0}^{N-1} f[n] e^{-j \frac{2\pi k}{N} n} \]

\[ f[n] = \sum_{k=0}^{N-1} F[k] e^{j \frac{2\pi k}{N} n} \]

Two dimensional DFT:

\[ F[k_r, k_c] = \frac{1}{RC} \sum_{r=0}^{R-1} \sum_{c=0}^{C-1} f[r, c] e^{-j \left( \frac{2\pi k_r}{R} r + \frac{2\pi k_c}{C} c \right)} \]

\[ f[r, c] = \sum_{k_r=0}^{R-1} \sum_{k_c=0}^{C-1} F[k_r, k_c] e^{j \left( \frac{2\pi k_r}{R} r + \frac{2\pi k_c}{C} c \right)} \]
# 2-D DFT: Properties

Just like in 1D, we can categorize and leverage various properties of the 2D DFT, for example:

<table>
<thead>
<tr>
<th>Property</th>
<th>$y[r, c]$</th>
<th>$Y[k_r, k_c]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linearity</td>
<td>$Ax_1[r, c] + Bx_2[r, c]$</td>
<td>$AX_1[k_r, k_c] + BX_2[k_r, k_c]$</td>
</tr>
<tr>
<td>Horizontal Flip</td>
<td>$x[r, C - c]$</td>
<td>$X[k_r, -k_c]$</td>
</tr>
<tr>
<td>Vertical Flip</td>
<td>$x[R - r, c]$</td>
<td>$X[-k_r, k_c]$</td>
</tr>
<tr>
<td>Spatial Shift (Circular)</td>
<td>$x[(r - r_0) \mod R, (c - c_0) \mod C]$</td>
<td>$e^{-j \frac{2\pi k_r}{R} r_0} e^{-j \frac{2\pi k_c}{C} c_0} X[k_r, k_c]$</td>
</tr>
<tr>
<td>Frequency Shift</td>
<td>$e^{j \frac{2\pi k_r}{R} r} e^{j \frac{2\pi k_c}{C} c} x[n]$</td>
<td>$X[k_r - k_r0, k_c - k_c0]$</td>
</tr>
</tbody>
</table>
The Dome

[Diagram of the Dome with labeled axes and a 2D contour plot.]

- The x-axis is labeled $k_r$ and the y-axis is labeled $k_c$. The plot shows a 2D contour plot with a color bar ranging from 0.0 to 1.0.
- The title of the plot is $|X(k_r, k_c)|$.

[Image of the dome building with additional architectural details added in the diagram.]
The Dome
The Dome
The Dome
The Dome

[Image of the Dome]

[Graph showing X[k_r, k_c] with k_c on the x-axis and k_r on the y-axis]

[Legend showing the color scale from 0.0 to 1.0]
The Dome
We can think about 2D filtering similarly to how we thought about filtering of 1D signals:

If the system is linear and shift-invariant, with a unit-sample response $h[r, c]$, then we can think of the output of the system in response to an arbitrary input $f[r, c]$ as a convolution: $(f * h)[r, c]$. 
2D Convolution

If a system is linear and shift-invariant, the response of the system to an input \( f[r, c] \) is the superposition of shifted and scaled versions of the unit-sample response:
2D Convolution

Convolution in space is equivalent to multiplication in frequency.

\[ f[r, c] = (f_a * f_b)[r, c] = \sum_{m_r=-\infty}^{\infty} \sum_{m_c=-\infty}^{\infty} f_a[m_r, m_c] f_b[r-m_r, c-m_c] \]

Discrete-Time Fourier Transform (all sums are over \(-\infty, \infty\)):

\[ F(\Omega_r, \Omega_c) = \sum_r \sum_c f[r, c] e^{-j\Omega_r r} e^{-j\Omega_c c} \]

\[ = \sum_r \sum_c \sum_{m_r} \sum_{m_c} f_a[m_r, m_c] f_b[r-m_r, c-m_c] e^{-j\Omega_r r} e^{-j\Omega_c c} \]

\[ = \sum_{m_r} \sum_{m_c} f_a[m_r, m_c] \sum_r \sum_c f_b[r-m_r, c-m_c] e^{-j\Omega_r r} e^{-j\Omega_c c} \]

\[ = \sum_{m_r} \sum_{m_c} f_a[m_r, m_c] \sum_{l_r} \sum_{l_c} f_b[l_r, l_c] e^{-j\Omega_x(l_r+m_r)} e^{-j\Omega_y(l_c+m_c)} \]

\[ = \sum_{m_r} \sum_{m_c} f_a[m_r, m_c] e^{-j\Omega_r m_r} e^{-j\Omega_c m_c} \sum_{l_r} \sum_{l_c} f_b[l_r, l_c] e^{-j\Omega_{lr} l_r} e^{-j\Omega_{lc} l_c} \]

\[ = F_a(\Omega_r, \Omega_c) F_b(\Omega_r, \Omega_c) \]
Example: Fishing

Consider the following image:

What output is produced by the following code?

```python
kernel = numpy.zeros((61, 61))
kernell[8, -16] = 1

fish = png_read('fish.png')
new = ifft2(fft2(fish) * fft2(kernel))
out = png_write(new, 'output.png')
```
Example: Fishing
Circular Convolution

1D: Multiplication of DFTs corresponds to **circular** convolution (and scaling by $1/N$) in time. Assume that $F[k]$ is the product of the DFTs of $f_1[n]$ and $f_2[n]$.

$$f[n] = \sum_{k=0}^{N-1} F[k] e^{j\frac{2\pi k}{N} n} = \sum_{k=0}^{N-1} F_1[k] F_2[k] e^{j\frac{2\pi k}{N} n}$$

$$= \sum_{k=0}^{N-1} F_1[k] \left( \frac{1}{N} \sum_{m=0}^{N-1} f_2[m] e^{-j\frac{2\pi k}{N} m} \right) e^{j\frac{2\pi k}{N} n}$$

$$= \frac{1}{N} \sum_{m=0}^{N-1} f_2[m] \left( \sum_{k=0}^{N-1} F_1[k] e^{j\frac{2\pi k}{N} (n-m)} \right)$$

$$= \frac{1}{N} \sum_{m=0}^{N-1} f_2[m] f_{1p}[n-m]$$

where $f_{1p}[n] = f_1[n \mod N]$ is a periodically extended version of $f_1[n]$. We refer to this as **circular** or **periodic** convolution:

$$\frac{1}{N} (f_1 \ast f_2)[n] \overset{\text{DFT}}{\iff} F_1[k] \times F_2[k]$$
Circular Convolution

\[ X_1[k] \times X_2[k] \xrightarrow{\text{DFT}} \frac{1}{N} (x_1 \odot x_2)[n] \]

where \( \odot \) denotes the **circular convolution** operator.

The result of circular convolution is equivalent to:

- a periodically-extended version (periodic in \( N \)) of the result of convolving the two signals:
  \[
  (x \odot h)[n] = \sum_{m=-\infty}^{\infty} (x * h)[n - mN]
  \]

- a convolution of one of the signals with a periodically-extended version of the other:
  \[
  (x \odot h)[n] = (x * h_p)[n], \text{ where } h_p[n] = \sum_{m=-\infty}^{\infty} h[n - mN]\]
2D Convolution: Conventional versus Circular

Similar in 2D.

Conventional convolution: direct form (space domain) or DTFT

Circular convolution: DFT
2D Convolution: Conventional versus Circular

The output of conventional convolution can be bigger than the input, while that of circular convolution aliases to the same size as the input.
Zero Padding

Differences between conventional and circular convolution can be reduced by padding the signal (1D or 2D) with a neutral border.

Here “neutral” is black.

In more natural images, a gray shade may introduce fewer artifacts. Could automatically choose an appropriate color by extending edge conditions.
Example: Fishing with Zero Pad
Example: Fishing with “Extended Edges”
Check Yourself!

Given the following image, what happens if we apply a filter that zeros out all the high frequencies in the image?
Check Yourself!

```python
from lib6003.fft import *
from lib6003.image import *

X = fft2(png_read('bluegill.png'))
N = X.shape[0]
assert N == X.shape[1]

for kr in range(-(N//2), N//2+1):
    for kc in range(-(N//2), N//2+1):
        if (kr**2 + kc**2)**0.5 > 25:
            X[kr, kc] = 0

show_image(ifft2(X))
```
Check Yourself!
Check Yourself!

The operation from the previous slide is equivalent to filtering:
\[ Y[k_r, k_c] = X[k_r, k_c] \times H_L[k_r, k_c], \]
where
\[ H_L[k_r, k_c] = \begin{cases} 
1 & \text{if } \sqrt{k_r^2 + k_c^2} \leq 25 \\
0 & \text{otherwise} 
\end{cases} \]

```python
X = fft2(png_read('bluegill.png'))
N = X.shape[0]
assert N == X.shape[1]

LPF = numpy.zeros_like(X)
for kr in range(-(N//2), N//2+1):
    for kc in range(-(N//2), N//2+1):
        if (kr**2 + kc**2)**0.5 <= 25:
            LPF[kr, kc] = 1

show_image(ifft2(X * LPF))
```

What is the equivalent spatial-domain operation?
Circular Convolution

Multiplying by that filter is equivalent to circular convolution with its spatial-domain equivalent. What does that look like?

show_image(ifft2(LPF))
Circular Convolution

Multiplying by that filter is equivalent to circular convolution with its spatial-domain equivalent. What does that look like?

`show_image(ifft2(LPF))`
Check Yourself!

Consider using the following filter instead:

\[
H_{L2}[k_r, k_c] = \begin{cases} 
\frac{1}{2} + \frac{1}{2} \cos \left( \pi \times \frac{\sqrt{k_r^2 + k_c^2}}{25} \right) & \text{if } \sqrt{k_r^2 + k_c^2} \leq 25 \\
0 & \text{otherwise}
\end{cases}
\]

\[M = 25\]

\[LPF2 = \text{numpy.zeros_like}(X)\]

\[\text{for kr in range}(-(N//2), N//2+1):\]
\[\quad \text{for kc in range}(-(N//2), N//2+1):\]
\[\quad \quad \text{dist} = (kr**2 + kc**2)**0.5\]
\[\quad \quad \text{if dist} \leq M:\]
\[\quad \quad \quad \text{LPF2}[kr, kc] = 0.5 + 0.5*\text{math.cos}(\text{math.pi} * \text{dist}/M)\]

How would you expect this output to compare to the result of the previous filter? Why?
Check Yourself!
Check Yourself!
Check Yourself: Comparing Filters

Filter 1

Filter 2
Imagine zeroing out the low frequencies instead. How will this image look? How does this filter relate to the previous filter in the frequency domain? In the spatial domain?
Check Yourself!

We can think of this new filter as:

\[ H_H[k_r, k_c] = 1 - H_L[k_r, k_c] = \begin{cases} 
1 & \text{if } \sqrt{k_r^2 + k_c^2} > 25 \\
0 & \text{otherwise}
\end{cases} \]

In the spatial domain, then, we have:
Check Yourself!

We can think of this new filter as:

\[
H_H[k_r, k_c] = 1 - H_L[k_r, k_c] = \begin{cases} 
1 & \text{if } \sqrt{k_r^2 + k_c^2} > 25 \\
0 & \text{otherwise}
\end{cases}
\]

In the spatial domain, then, we have:

\[
h_H[r, c] = RC\delta[r, c] - h_L[r, c]
\]
HPF
We can reduce the ringing artifacts by using $1 - H_{L2}[k_r, k_c]$ instead:
An Interesting Optical Illusion

A *hybrid image* is created by combining the low frequencies from one image with the high frequencies of another. Consider combining the low frequencies from the image on the left with the high frequencies from the image on the right:
Hybrid Image

```python
ein = fft2(png_read('lec11a_code/einstein.png'))
mar = fft2(png_read('lec11a_code/marilyn.png'))

ein = (1-LPF2) * ein
mar = (LPF2) * mar

show_image(ifft2(ein + mar))
```
Hybrid Image
Deconvolution Primer

The image on the right was created by circularly convolving the image on the left with *something*. What was that something?
Deconvolution Results

\[
\text{ifft2}(\text{fft2}(\text{png\_read}('\text{conv.png}'))/\text{fft2}(\text{png\_read}('\text{zebra.png}')))
\]
Deconvolution Results

\[ \text{ifft2(fft2(png\_read('conv.png'))/fft2(png\_read('zebra.png')))} \]
Summary

Today:
- 2D frequency representations
- 2D filtering

Recitation:
- More on 2D convolution/filtering

Thursday:
- Deconvolution

Next week:
- Quiz week!