2D Fourier Transforms
- Directionality and rotation
- Magnitudes of Fourier transforms
- Phases of Fourier transforms
- 2D Filtering

April 28, 2020
2D signal processing builds on simple extensions of 1D.

Signals: \( t \rightarrow x, y \)

Transforms: \( \int \rightarrow \int \int \)

Basis Functions: \( e^{j\omega t} \rightarrow e^{j(\omega_xx + \omega_yy)} \)

Today: intuition for 2D basis functions and implications for 2D transforms.
2D Discrete Fourier Transform

As in 1D, the DFT is most convenient for computations.

**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:**

\[
F[k_x, k_y] = \frac{1}{N_x N_y} \sum_{n_x=0}^{N_x-1} \sum_{n_y=0}^{N_y-1} f[n_x, n_y] \, e^{-j \left( \frac{2\pi k_x}{N_x} n_x + \frac{2\pi k_y}{N_y} n_y \right)}
\]

\[
f[n_x, n_y] = \sum_{k_x=0}^{N_x-1} \sum_{k_y=0}^{N_y-1} F[k_x, k_y] \, e^{j \left( \frac{2\pi k_x}{N_x} n_x + \frac{2\pi k_y}{N_y} n_y \right)}
\]

2D domain; double sums; 2D kernel function.
2D Discrete Fourier Transform

Alternatively, implement a 2D DFT as a sequence of 1D DFTs.

\[
F[k_x, k_y] = \frac{1}{N_x N_y} \sum_{n_x=0}^{N_x-1} \sum_{n_y=0}^{N_y-1} f[n_x, n_y] e^{-j\left(\frac{2\pi k_x}{N_x} n_x + \frac{2\pi k_y}{N_y} n_y\right)}
\]

\[
= \frac{1}{N_y} \sum_{n_y=0}^{N_y-1} \left( \frac{1}{N_x} \sum_{n_x=0}^{N_x-1} f[n_x, n_y] e^{-j\frac{2\pi k_x}{N_x} n_x} \right) e^{-j\frac{2\pi k_y}{N_y} n_y}
\]

first take DFTs of rows

then take DFTs of resulting columns

Could just as well start with columns and then do rows.
Example: Find the DFT of a 2D unit sample.
Example: Find the DFT of a 2D unit sample.
Example: Find the DFT of a 2D unit sample.

- **Magnitude**: $f[n_x, n_y]$
- **Angle**: $f[n_x, n_y]$
Example: Find the DFT of a 2D unit sample.
2D Discrete Fourier Transform

Example: Find the DFT of a 2D unit sample.
2D Discrete Fourier Transform

Example: Find the DFT of a 2D unit sample.
2D Discrete Fourier Transform

Example: Find the DFT of a 2D unit sample.
2D Discrete Fourier Transform

Example: Find the DFT of a 2D unit sample.
2D Discrete Fourier Transform

Example: Find the DFT of a 2D unit sample.

Magnitude

$|f[n_x, n_y]|$

$n_x$

$n_y$

Angle

$\angle f[n_x, n_y]$

$n_x$

$n_y$

$|F[k_x, n]|$

$k_x$

$n$

DFT(rows)
2D Discrete Fourier Transform

Example: Find the DFT of a 2D unit sample.
Example: Find the DFT of a 2D unit sample.

- **Magnitude**
  - $f[n_x, n_y]$

- **Angle**

- **DFT(rows)**
  - $n$
  - $k_x$

- **DFT(rows)**
  - $n$
  - $k_x$
Example: Find the DFT of a 2D unit sample.
2D Discrete Fourier Transform

Example: Find the DFT of a 2D unit sample.
2D Discrete Fourier Transform

Example: Find the DFT of a 2D unit sample.
2D Discrete Fourier Transform

Example: Find the DFT of a 2D unit sample.
Example: Find the DFT of a 2D unit sample.
Example: Find the DFT of a 2D unit sample.
2D Discrete Fourier Transform

Example: Find the DFT of a 2D unit sample.
Example: Find the DFT of a 2D unit sample.
Example: Find the DFT of a 2D unit sample.
Example: Find the DFT of a 2D unit sample.
2D Discrete Fourier Transform

Example: Find the DFT of a 2D unit sample.
2D Discrete Fourier Transform

Example: Find the DFT of a 2D unit sample.
2D Discrete Fourier Transform

The DFT of a 2D unit sample is a constant.
2D Discrete Fourier Transform

The DFT of a constant is a 2D unit sample.
The 2D Discrete Fourier Transform (DFT) of a vertical line is a horizontal line.
2D Discrete Fourier Transform

The DFT of a horizontal line is a vertical line.
Rotation of Images

Example: Find the DFT of a cosine wave.
Rotation of Images

Example: Find the DFT of a rotated cosine wave.
Rotation of Images

Example: Find the DFT of a rotated cosine wave.
Rotation of Images

Example: Find the DFT of a rotated cosine wave.
Rotation of Images

Example: Find the DFT of a cosine wave.

Magnitude
\[ f[n_x, n_y] \]

Angle

DFT (rows)

Frequency

\[ F[k_x, k_y] \]
Rotation of Images

Rotating an image rotates its Fourier transform by the same angle.

Rewrite the 2D CTFT relation in polar form.

\[ F(\omega_x, \omega_y) = \int \int f(x, y) e^{-j(\omega_xx + \omega_yy)} \, dx \, dy \]

Express points \((x, y)\) in space as \((r, \theta)\) and points \((\omega_x, \omega_y)\) in the frequency plane as \((\omega, \phi)\).

\[
\omega_xx + \omega_yy = \underbrace{\omega \cos \phi}_\omega \underbrace{r \cos \theta}_x + \underbrace{\omega \sin \phi}_\omega \underbrace{r \sin \theta}_y
\]

\[
= \omega r (\cos \phi \cos \theta + \sin \phi \sin \theta) = \omega r \cos (\phi - \theta)
\]

Then

\[ F_r(\omega, \phi) = \int \int f_r(r, \theta) e^{-j\omega r \cos(\phi - \theta)} \, r \, dr \, d\theta \]

where \(f_r(r, \theta)\) and \(F_r(\omega, \phi)\) are polar equivalents of \(f(x, y)\) and \(F(\omega_x, \omega_y)\).
Rotation of Images

Rotating an image rotates its Fourier transform by the same angle.

\[ F_r(\omega, \phi) = \int \int f_r(r, \theta) e^{-j\omega r \cos(\phi-\theta)} r \, dr \, d\theta \]

If

\[ f_1(r, \theta) \overset{\text{CTFT}}{\Rightarrow} F_1(\omega, \phi) \]

and

\[ f_2(r, \theta) = f_1(r, \theta-\psi) \]

Then

\[ F_2(\omega, \phi) = \int \int f_2(r, \theta) e^{-j\omega r \cos(\phi-\theta)} r \, dr \, d\theta \]

\[ = \int \int f_1(r, \theta-\psi) e^{-j\omega r \cos(\phi-\theta)} r \, dr \, d\theta \]

\[ = \int \int f_1(r, \lambda) e^{-j\omega r \cos(\phi-\lambda-\psi)} r \, dr \, d\lambda \]

\[ = F_1(\omega, \phi - \psi) \]
Calculating DFTs is most efficient in NumPy (Numerical Python).

- NumPy arrays are **homogeneous**: their elements are of the same type
- Numpy operators (+, -, abs, .real, .imag) combine elements to create new arrays. e.g., (f+g)[n] is f[n]+g[n].
- 2D Numpy arrays can be **indexed by tuples**: e.g., f[r,c] = f[r][c].
- 2D Numpy arrays support **negative indices**: e.g., f[-1] = f[len(a)-1]
- 2D indices address **row then column**.

```
f[0, 0]  f[0, 1]  f[0, 2]  f[0, 3]  ...
f[1, 0]  f[1, 1]  f[1, 2]  f[1, 3]  ...
f[2, 0]  f[2, 1]  f[2, 2]  f[2, 3]  ...
f[3, 0]  f[3, 1]  f[3, 2]  f[3, 3]  ...
```

NumPy indexing is consistent with **linear algebra** (row first then column with rows increasing downward and columns increasing to the right). But it differs from **physical mathematics** ($x$ then $y$ with $x$ increasing to the right and $y$ increasing upward). You may do calculations either way, but row,column is often less confusing.
Numpy Example

Make a white square on a black background.

```python
import numpy
from lib6003.image import show_image
f = numpy.zeros((64,64))
for r in range(16,48):
    for c in range(16,48):
        f[r,c] = 1
show_image(f,zero_loc='topleft')
```
Numpy Example

Find the 2D DFT of the square.

```python
import numpy
from lib6003.image import show_image
from lib6003.fft import fft2

F = fft2(f)
show_image(numpy.abs(F), zero_loc='topleft')
```
Big and Small
Triangle

What are the dominant features of the magnitude of the DFT of a triangle?
What are the dominant features of the magnitude of the DFT of a triangle?

The DFT has three nearly linear features, one for each edge of the triangle. Lines in the frequency domain are perpendicular to those in space domain.
What are the dominant features of the DFT magnitude of an ocean view?
What are the dominant features of the DFT magnitude of an ocean view?

The horizontal features in the ocean view show up as a strong vertical line in the DFT.
Trees

What are the dominant features of the DFT magnitude of these trees?
Trees

What are the dominant features of the DFT magnitude of these trees?

Now there is a strong horizontal line in the DFT.
What are the dominant features of the DFT magnitude of this rose?
What are the dominant features of the DFT magnitude of this rose?

Dominant frequencies are close to $k_r = k_c = 0$, i.e., low frequencies.
What are the dominant features of the DFT magnitude of the moon?
What are the dominant features of the DFT magnitude of the moon?

Large distribution of frequencies. Concentration along $r = c$ axis due to illumination from upper left.
Bricks

What are the dominant features of the DFT magnitude of this brick wall?
What are the dominant features of the DFT magnitude of this brick wall?

- Strong horizontal “layer” in space → strong vertical line in frequency.
- Strong vertical features but broken up periodically → periodicity in frequency.
Fingerprint

What are the dominant features of the DFT magnitude of this fingerprint?
What are the dominant features of the DFT magnitude of this fingerprint?

Dominant frequencies form a ring at $\Omega \approx \frac{\pi}{3}$. 
Magnitude and Phase

So far, we have only considered magnitude. Does phase matter?

There is clearly structure in the magnitude; phase looks random.
Magnitude and Phase

Zeroing out the phase has an enormous impact on the image.

Phase is clearly important.
Magnitude and Phase

Flattening the magnitude has a big effect.

But the image is still recognizable!
Magnitude and Phase

Substituting the magnitude from a different image has a big effect.

But the boat is recognizable. What magnitude was used?
The magnitude for the previous image was taken from this image.
Magnitude and Phase

Here is the original again.

The boat is recognizable. What magnitude was used?
Magnitude and Phase

Here is the hybrid image: mandrill magnitude + boat phase.

Phase is very important in images.
Discrete Fourier Series of Sounds

We previously looked at Fourier representations for sounds. Phase played a minor role in auditory perception. These signals have the same magnitudes but different phases.

But they all sound very similar to each other.
Why are images so sensitive to phase?

All Fourier components must have correct phase to preserve an edge. Changing the phase of just one component can have a drastic effect on an image.
Auditory Perception of Phase

Why are we insensitive to the phase of components of sound? Different frequencies are processed in different regions of the cochlea, with little sensitivity to changes in phase across frequency regions.

However, we are very sensitive to binaural phase differences at a given frequency.
2D Filtering

How can we remove the high frequencies from this image.

One method is to transform, zero out the high-frequency components, and inverse transform.
Transform, zero out the high-frequency components, and inverse transform.

```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))
```
2D Filtering

Transform, zero out the high-frequency components, and inverse transform.

Where did the ripples come from?
Zeroing out frequency components is equivalent to filtering by
\[
H_L[k_r, k_c] = \begin{cases} 
1 & \text{if } \sqrt{k_r^2 + k_c^2} \leq 25 \\
0 & \text{otherwise}
\end{cases}
\]

Find the 2D unit-sample response of this filter.

\[
X = \text{fft2(png_read(}\text{\textquoteleft}\text{bluegill.png}\text{\textquoteright}))
\]

\[
N = X.\text{shape}[0]
\]

\[
\text{assert } N == X.\text{shape}[1]
\]

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

\[
\text{for kr in range(-}(N//2), N//2+1):}
\]
\[
\text{for kc in range(-}(N//2), N//2+1):}
\]
\[
\text{if (kr**2 + kc**2)**0.5} \leq 25:
\]
\[
\text{LPF[kr, kc] = 1}
\]

\[
\text{show_image(ifft2(X * LPF))}
\]
2D Filtering

Find the 2D unit-sample response of this filter.

```
show_image(ifft2(LPF))
```

The step changes in $|H_L[k_r, k_c]|$ generated overshoot: Gibb’s phenomenon.
Consider using the following filter, which is a circularly symmetric version of the Hann window.

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

\[
M = 25
\]

```python
LPF2 = numpy.zeros_like(X)
for kr in range(-(N//2), N//2+1):
    for kc in range(-(N//2), N//2+1):
        dist = (kr**2 + kc**2)**0.5
        if dist <= M:
            LPF2[kr, kc] = 0.5 + 0.5*math.cos(math.pi * dist/M)
```
2D Filtering

Ripples are gone.
2D Filtering

Ripples are gone.
Comparing Filters

Filter 1

Filter 2
High-Pass Filtering

Use the same approaches to implement a high-pass filter.

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

In the spatial domain, then, we have:

\[ h_H[r, c] = RC\delta[r, c] - h_L[r, c] \]
High-Pass Filtering

Not surprisingly, results show the same rippling effect seen in LPF.
High-Pass Filtering

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 (left) with the high frequencies of another (right).
Hybrid Image

```
hybrid_image = 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

Look at this image with your eyes about a foot away from the screen. Then look again from a distance of six feet. Any difference?
Summary

2D Fourier basis functions explicitly code **spatial direction**. The directionality is reflected in both the space and frequency domains, although in perpendicular directions.

**Rotating** an image rotates its transform.

Many features of an image (such as the orientations of structures) are apparent in the **magnitude** of the Fourier transform, but the **phase** of the Fourier transform is crucial to representing sharp edges.

Sharp transitions between passbands and stopbands produce **overshoot** (Gibb's phenomenon) in 2D filters. This ringing shows up as **ripples** in processed images.