Capturing what’s important

$x_1$ $x_0$

2nd principal component

1st principal component

[Images of scatter plots and faces]
Fast vs. slow changes
A nice set of basis

Teases away fast vs. slow changes in the image.

This change of basis has a special name…
Jean Baptiste Joseph Fourier (1768-1830)

had crazy idea (1807):

*Any* periodic function can be rewritten as a weighted sum of sines and cosines of different frequencies.

Don’t believe it?

- Neither did Lagrange, Laplace, Poisson and other big wigs
- Not translated into English until 1878!

But it’s true!

- called Fourier Series
A sum of sines

Our building block:

$$A \sin(\omega x + \phi)$$

Add enough of them to get any signal \(f(x)\) you want!

How many degrees of freedom?

What does each control?

Which one encodes the coarse vs. fine structure of the signal?

\[ f(\text{target}) = f_1 + f_2 + f_3 \ldots + f_n + \ldots \]
Fourier Transform

We want to understand the frequency $\omega$ of our signal. So, let’s reparametrize the signal by $\omega$ instead of $x$:

$$f(x) \rightarrow \text{Fourier Transform} \rightarrow F(\omega)$$

For every $\omega$ from 0 to inf, $F(\omega)$ holds the amplitude $A$ and phase $\phi$ of the corresponding sine $A \sin(\omega x + \phi)$

- How can $F$ hold both? Complex number trick!

$$F(\omega) = R(\omega) + iI(\omega)$$

$$A = \pm \sqrt{R(\omega)^2 + I(\omega)^2}$$

$$\phi = \tan^{-1} \frac{I(\omega)}{R(\omega)}$$

We can always go back:

$$F(\omega) \rightarrow \text{Inverse Fourier Transform} \rightarrow f(x)$$
example: \( g(t) = \sin(2pf \ t) + (1/3)\sin(2p(3f) \ t) \)
Time and Frequency

example: \( g(t) = \sin(2\pi f t) + \frac{1}{3}\sin(2\pi (3f) t) \)
Frequency Spectra

example: \( g(t) = \sin(2pf t) + \frac{1}{3}\sin(2p(3f) t) \)
Frequency Spectra

Usually, frequency is more interesting than the phase.
Frequency Spectra
Frequency Spectra
Frequency Spectra

= +

=
Frequency Spectra
Frequency Spectra

\[ A \sum_{k=1}^{\infty} \frac{1}{k} \sin(2\pi kt) \]
Frequency Spectra

(a) Frequency spectrum of a periodic signal.

(b) Frequency spectrum of a square wave.

(c) Frequency spectrum of a random signal.
FT: Just a change of basis

\[ M \ast f(x) = F(\omega) \]
IFT: Just a change of basis

\[ M^{-1} \ast F(\omega) = f(x) \]
Finally: Scary Math

Fourier Transform: \[ F(\omega) = \int_{-\infty}^{+\infty} f(x)e^{-i\omega x} \, dx \]

Inverse Fourier Transform: \[ f(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega)e^{i\omega x} \, d\omega \]
Finally: Scary Math

Fourier Transform: \( F(\omega) = \int_{-\infty}^{+\infty} f(x)e^{-i\omega x} \, dx \)

Inverse Fourier Transform: \( f(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega)e^{i\omega x} \, d\omega \)

...not really scary: \( e^{i\omega x} = \cos(\omega x) + i \sin(\omega x) \)

is hiding our old friend: \( A\sin(\omega x + \phi) \)

phase can be encoded by sin/cos pair

\[ P \cos(x) + Q \sin(x) = A \sin(x + \phi) \]

\[ A = \pm \sqrt{P^2 + Q^2} \]

\[ \phi = \tan^{-1}\left(\frac{P}{Q}\right) \]

So it’s just our signal \( f(x) \) times sine at frequency \( \omega \)
Extension to 2D

in Matlab, check out: imagesc(log(abs(fftshift(fft2(im)))));
2D FFT transform
Man-made Scene
Can change spectrum, then reconstruct
Most information in at low frequencies!
Campbell-Robson contrast sensitivity curve

We don’t resolve high frequencies too well…
… let’s use this to compress images… JPEG!
Lossy Image Compression (JPEG)

Block-based Discrete Cosine Transform (DCT)
Using DCT in JPEG

A variant of discrete Fourier transform

- Real numbers
- Fast implementation

Block size

- small block
  - faster
  - correlation exists between neighboring pixels
- large block
  - better compression in smooth regions
Using DCT in JPEG

The first coefficient $B(0,0)$ is the DC component, the average intensity.

The top-left coeffs represent low frequencies, the bottom right – high frequencies.
Image compression using DCT

DCT enables image compression by concentrating most image information in the low frequencies.

Loose unimportant image info (high frequencies) by cutting $B(u,v)$ at bottom right.

The decoder computes the inverse DCT – IDCT.

- Quantization Table

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>7</td>
<td>9</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>9</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
<td>25</td>
</tr>
<tr>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
<td>25</td>
<td>27</td>
</tr>
<tr>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
<td>25</td>
<td>27</td>
<td>29</td>
</tr>
<tr>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
<td>25</td>
<td>27</td>
<td>29</td>
<td>31</td>
</tr>
</tbody>
</table>
JPEG compression comparison

89k  12k