Image Filtering, Edges and Image Representation

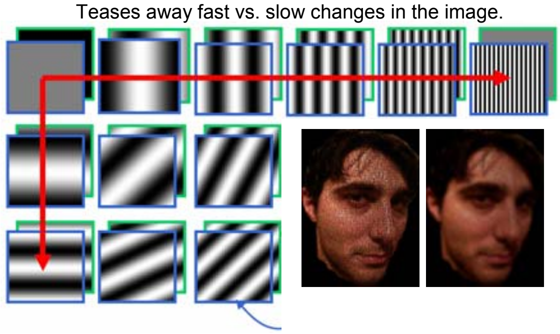
Req. reading:
–Chapter 7, 9.2 F&P
–Adelson, Simoncelli and Freeman (handout online)
Opt. reading:
–Horn 7 & 8
–FP 8

February 19, 2008

Capturing what's important

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:

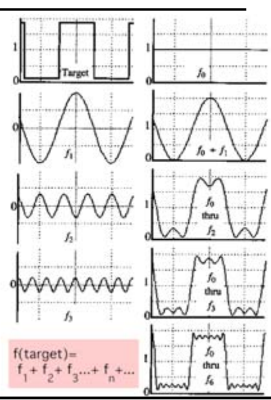
$$A\sin(\omega x + \phi)$$

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?



Fourier Transform

We want to understand the frequency ω of our signal. So, let's reparametrize the signal by ω instead of *x*:

f(x) ⟶ Fourier Transform ⟶ ***F(ω)***

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

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

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

$$A = \pm\sqrt{R(\omega)^2 + I(\omega)^2} \qquad \phi = \tan^{-1}\frac{I(\omega)}{R(\omega)}$$

We can always go back:

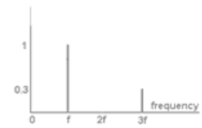
F(ω) ⟶ Inverse Fourier Transform ⟶ ***f(x)***

Time and Frequency

example : $g(t) = \sin(2\pi f\, t) + (1/3)\sin(2\pi(3f\, t))$

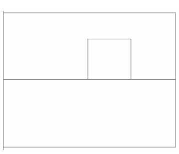
Frequency Spectra

example : $g(t) = \sin(2\pi f\, t) + (1/3)\sin(2\pi(3f)\, t)$

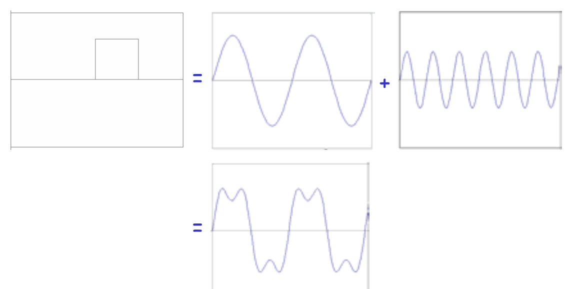


Frequency Spectra

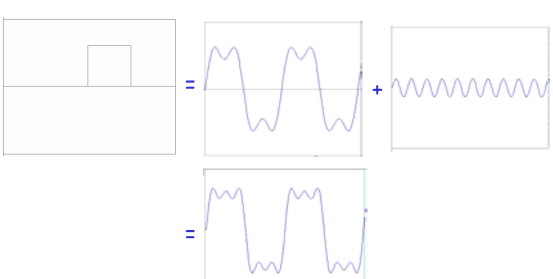
Usually, frequency is more interesting than the phase



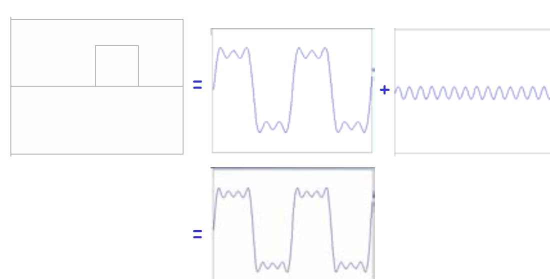
Frequency Spectra



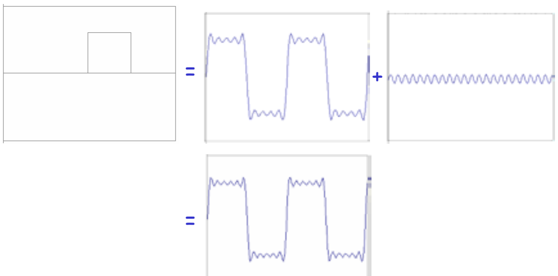
Frequency Spectra



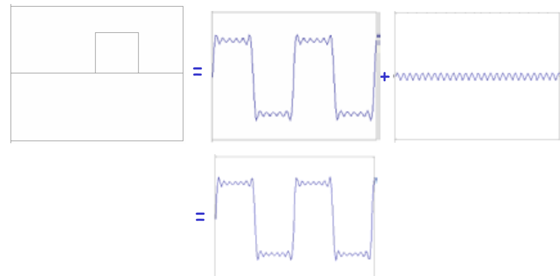
Frequency Spectra



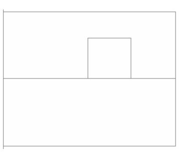
Frequency Spectra



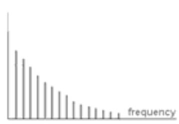
Frequency Spectra



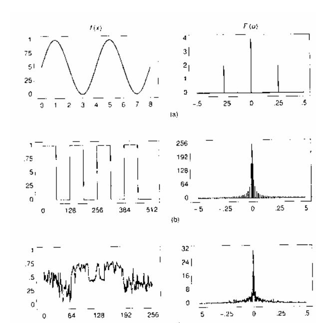
Frequency Spectra



$= A\sum_{k=1}^{\infty}\frac{1}{k}\sin(2\pi kt)$

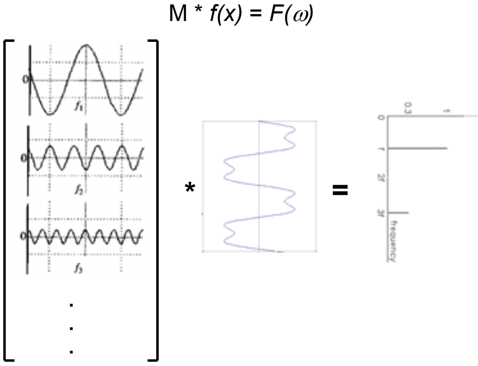


Frequency Spectra



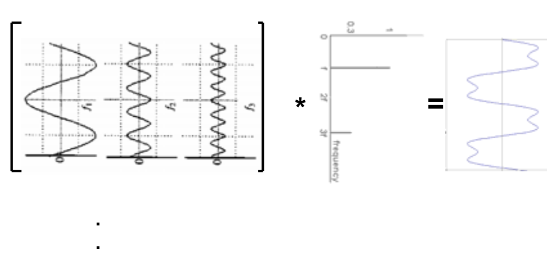
FT: Just a change of basis

M * $f(x) = F(\omega)$



IFT: Just a change of basis

$M^{-1} * F(\omega) = f(x)$



Finally: Scary Math

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

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

Finally: Scary Math

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

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

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

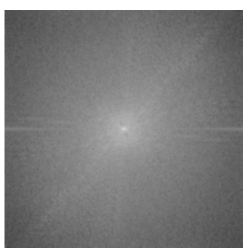
is hiding our old friend: $A\sin(\omega x + \phi)$

phase can be encoded by sin/cos pair $\longrightarrow$ $P\cos(x) + Q\sin(x) = A\sin(x+\phi)$

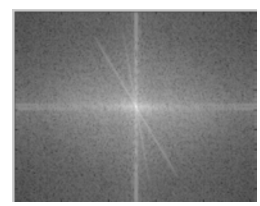
$A = \pm\sqrt{P^2 + Q^2} \qquad \phi = \tan^{-1}\left(\frac{P}{Q}\right)$

So it's just our signal *f(x)* times sine at frequency ω

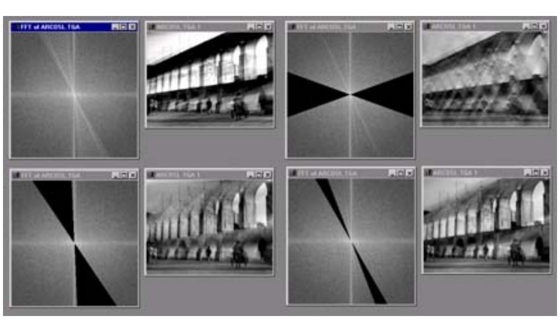
2D FFT transform



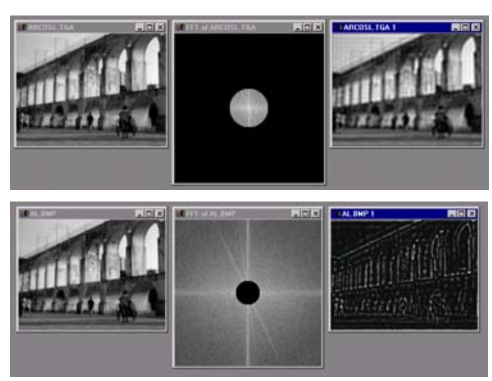
Man-made Scene



Can change spectrum, then reconstruct



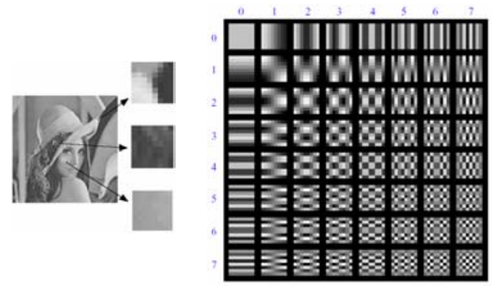
Most information in at low frequencies!



Application to image compression

- (compression is about hiding differences from the true image where you can't see them).

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

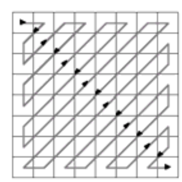


Image compression using DCT

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

3	5	7	9	11	13	15	17
5	7	9	11	13	15	17	19
7	9	11	13	15	17	19	21
9	11	13	15	17	19	21	23
11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29
17	19	21	23	25	27	29	31

JPEG compression comparison

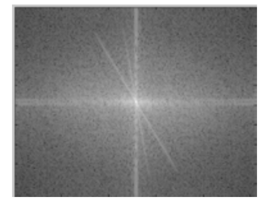
89k

12k

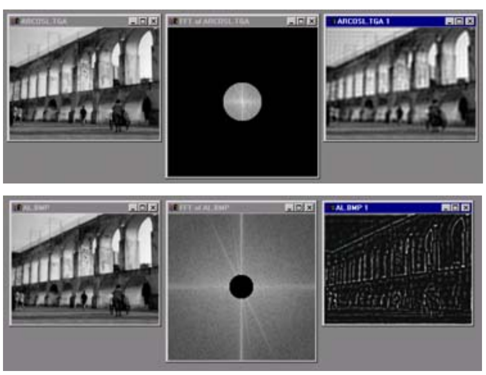
Convolution and Edge Detection

Reading: FP 8

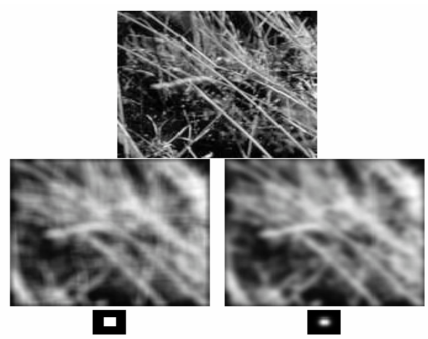
Man-made Scene



Most information in at low frequencies!



Mean vs. Gaussian filtering



Gaussian filtering

A Gaussian kernel gives less weight to pixels further from the center of the window

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

$F[x, y]$

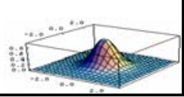
$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$H[u, v]$

This kernel is an approximation of a Gaussian function:

$$h(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



The Convolution Theorem

- The Fourier transform of the convolution of two functions is the product of their Fourier transforms

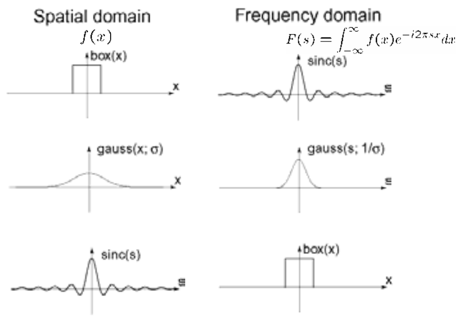
$$\mathrm{F}[g * h] = \mathrm{F}[g]\mathrm{F}[h]$$

- The inverse Fourier transform of the product of two Fourier transforms is the convolution of the two inverse Fourier transforms

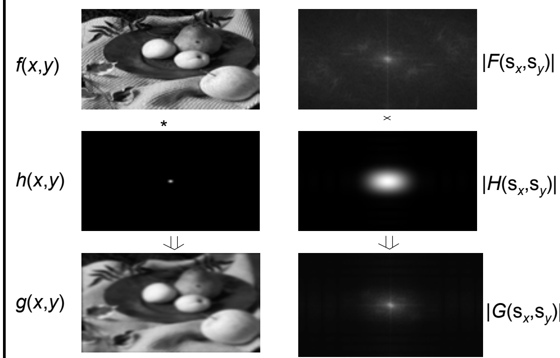
$$\mathrm{F}^{-1}[gh] = \mathrm{F}^{-1}[g] * \mathrm{F}^{-1}[h]$$

- **Convolution** in spatial domain is equivalent to **multiplication** in frequency domain!

Fourier Transform pairs



2D convolution theorem example



Edges in images

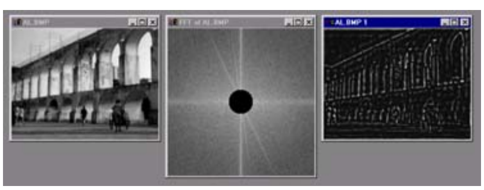
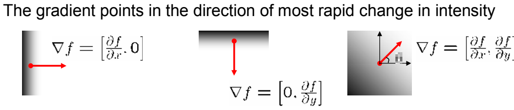


Image gradient

The gradient of an image:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

The gradient points in the direction of most rapid change in intensity



The gradient direction is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- how does this relate to the direction of the edge? *perpendicular*

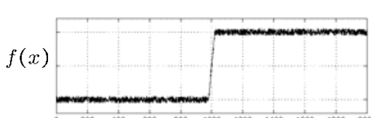
The *edge strength* is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

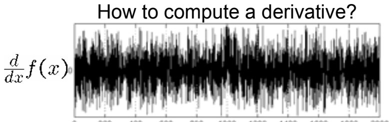
Effects of noise

Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

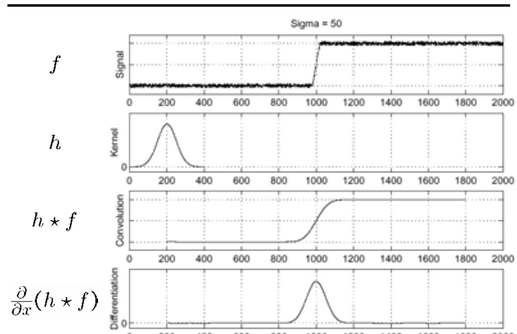


How to compute a derivative?



Where is the edge?

Solution: smooth first



Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem of convolution

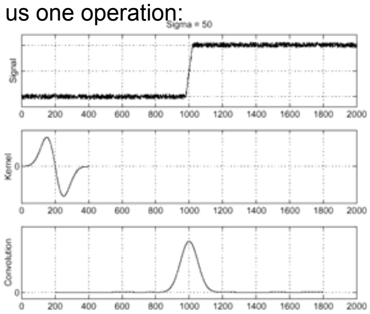
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x} h) \star f$$

This saves us one operation:

f

$\frac{\partial}{\partial x} h$

$(\frac{\partial}{\partial x} h) \star f$



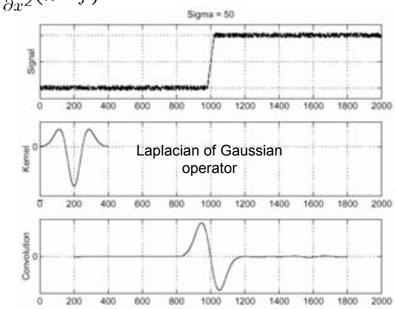
Laplacian of Gaussian

Consider $\frac{\partial^2}{\partial x^2}(h \star f)$

f

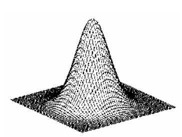
$\frac{\partial^2}{\partial x^2} h$

$(\frac{\partial^2}{\partial x^2} h) \star f$



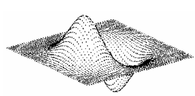
Where is the edge? Zero-crossings of bottom graph

2D edge detection filters



Gaussian

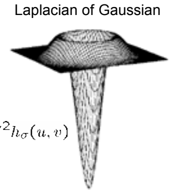
$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$



derivative of Gaussian

$\frac{\partial}{\partial x} h_\sigma(u,v)$

Laplacian of Gaussian



$\nabla^2 h_\sigma(u,v)$

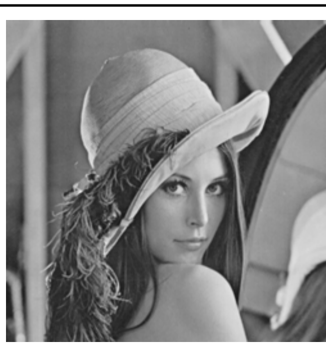
∇^2 is the **Laplacian** operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

MATLAB demo

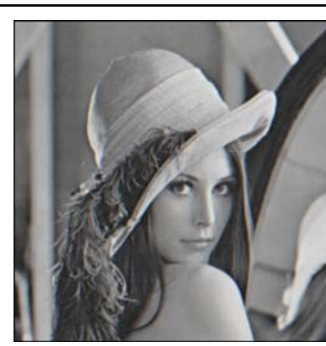
```
g = fspecial('gaussian',15,2);
imagesc(g)
surfl(g)
gclown = conv2(clown,g,'same');
imagesc(conv2(clown,[-1 1],'same'));
imagesc(conv2(gclown,[-1 1],'same'));
dx = conv2(g,[-1 1],'same');
imagesc(conv2(clown,dx,'same'));
lg = fspecial('log',15,2);
lclown = conv2(clown,lg,'same');
imagesc(lclown)
imagesc(clown + .2*lclown)
```

What does blurring take away?



original

What does blurring take away?



smoothed (5x5 Gaussian)

Edge detection by subtraction

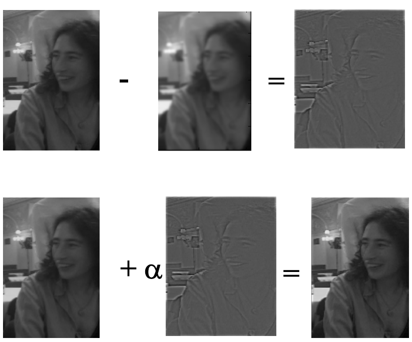
Why does this work?

smoothed – original

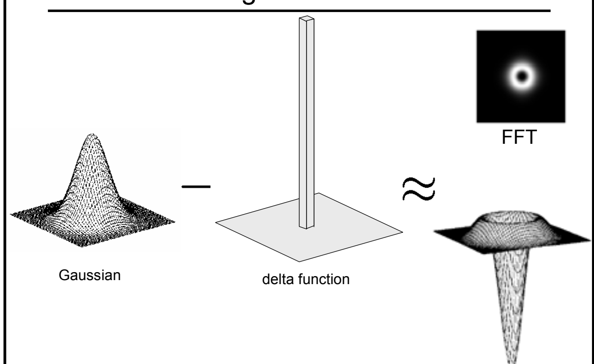
High-Pass filter

smoothed – original

Unsharp Masking

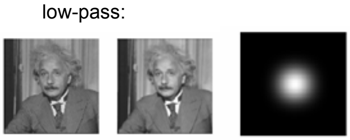


Gaussian - image filter



Low-pass, Band-pass, High-pass filters

low-pass:



High-pass / band-pass: