COMP 9517 Computer Vision

Frequency Domain Techniques

Why does a lower resolution image still make sense to us? What do we lose?

Image: http://www.flickr.com/photos/igorms/136916757/ Slide: Hoiem

Jean Baptiste Joseph Fourier (1768-1830)

had crazy idea (1807):

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

...the manner in which the author arrives at these equations is not exempt of difficulties and...his analysis to integrate them still leaves something to be desired on the score of generality and even rigour.

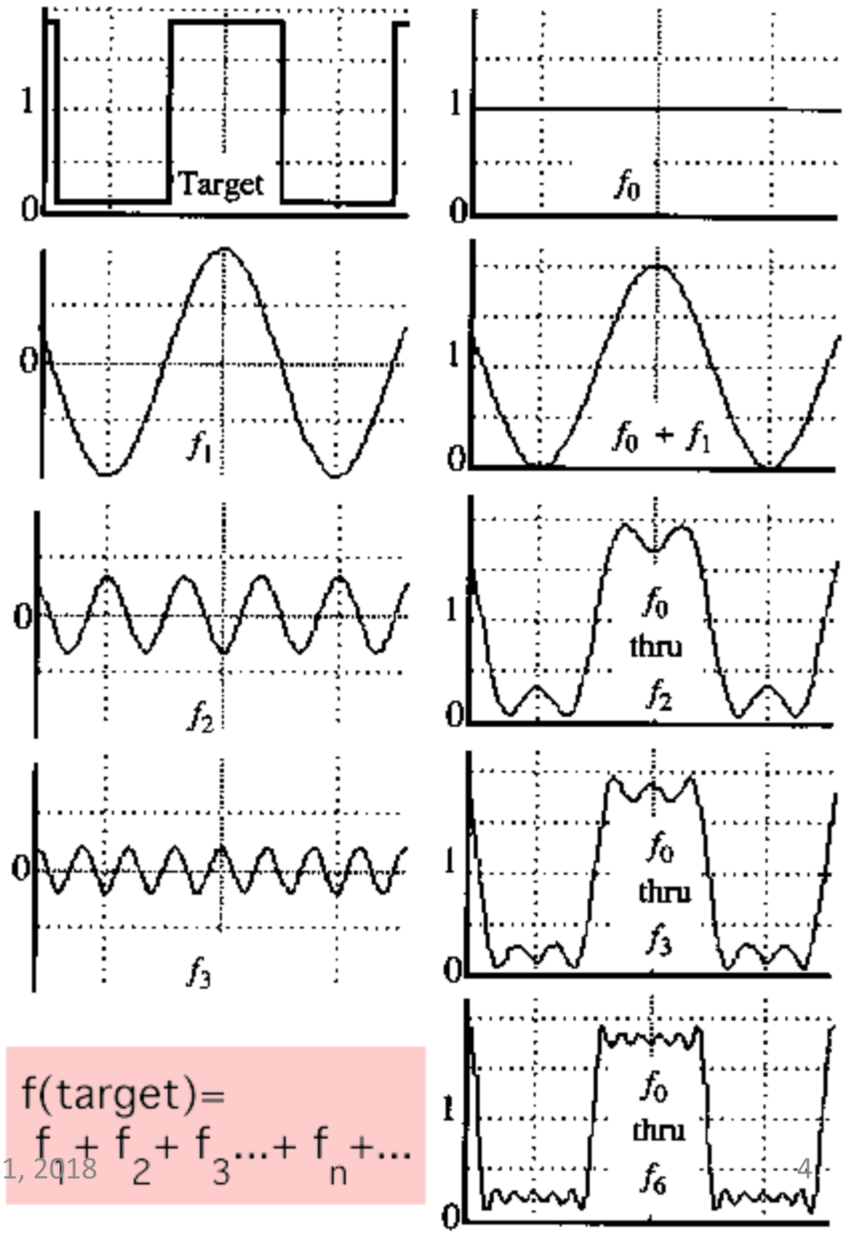
- Don't believe it?
  - Neither did Lagrange, Laplace, Poisson and other big wigs
  - Not translated into English until 1878!
- But it's (mostly) true!
  - called Fourier Series
  - there are some subtle restrictions

A sum of sines

Our building block:

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

Add enough of them to get any signal *g(x)* you want!



f(target)= $f_1 + f_2 + f_3 \ldots + f_n + \ldots$

Frequency Versus Spatial Domain

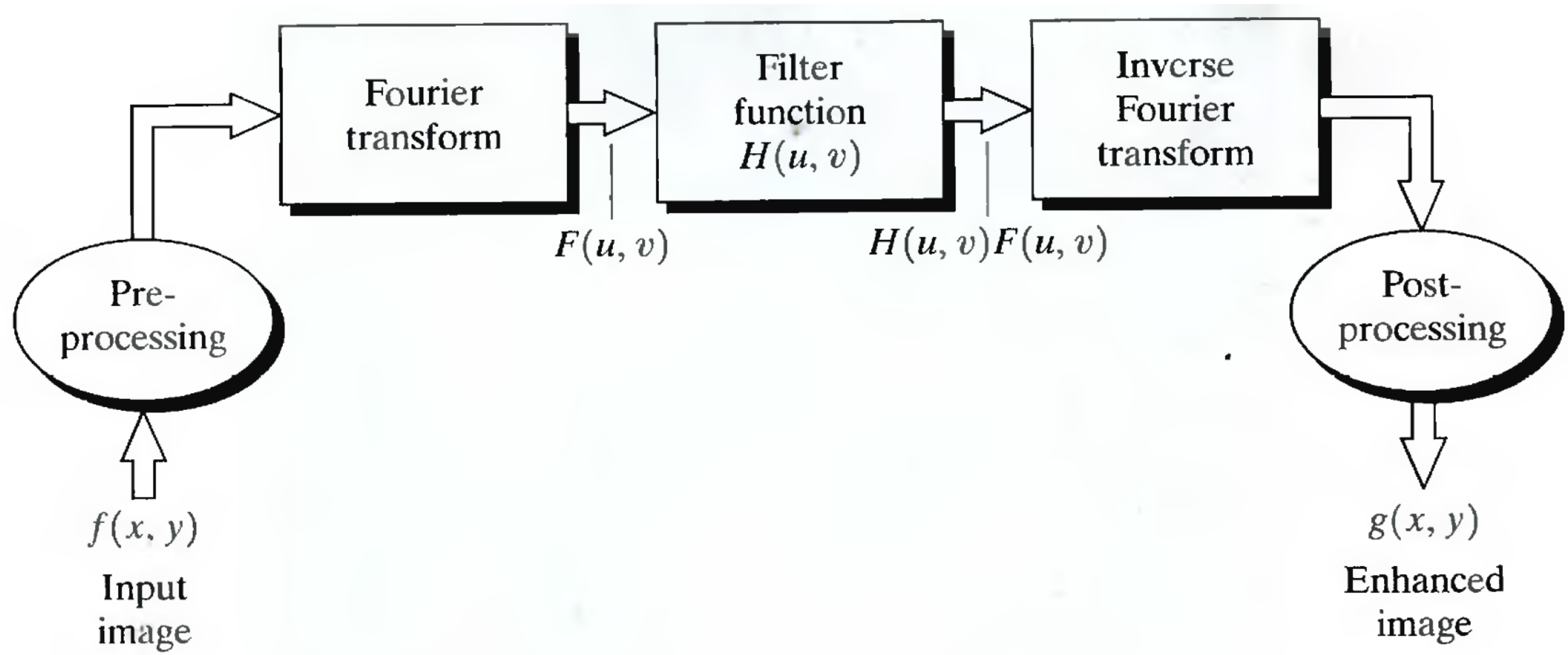
- Spatial domain
  - The image plane itself
  - Direct manipulation of pixels
  - Changes in pixel position correspond to changes in the scene
- Frequency domain
  - Fourier transform of an image
  - Directly related to rate of changes in the image
  - Changes in pixel position correspond to changes in the spatial frequency

Frequency Domain Overview

- Frequency in image
  - High frequencies correspond to pixel values that change rapidly across the image
  - Low frequency components correspond to large scale features in the image
- Frequency domain
  - Defined by values of the Fourier transform and its frequency variables (u, v)

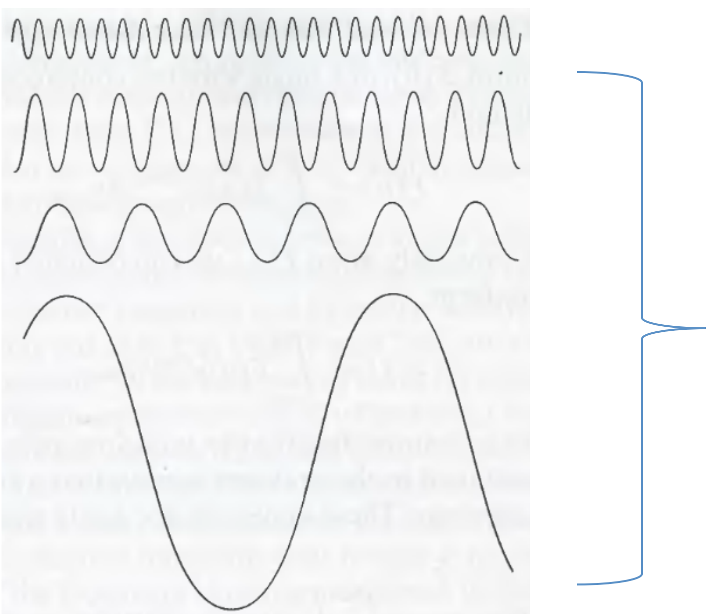
Frequency Domain Overview

- Frequency domain processing

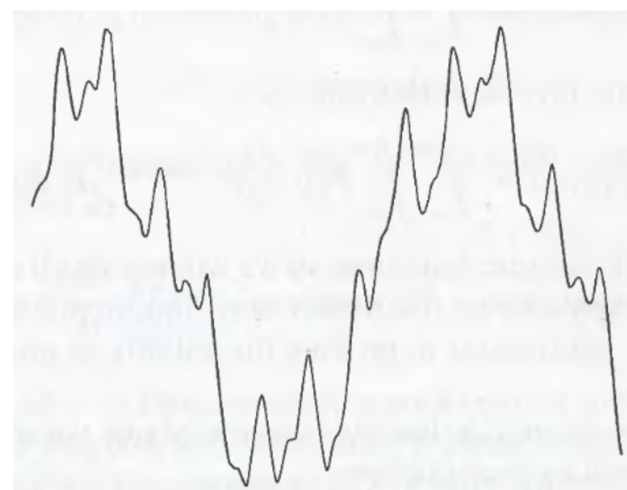


Fourier Series

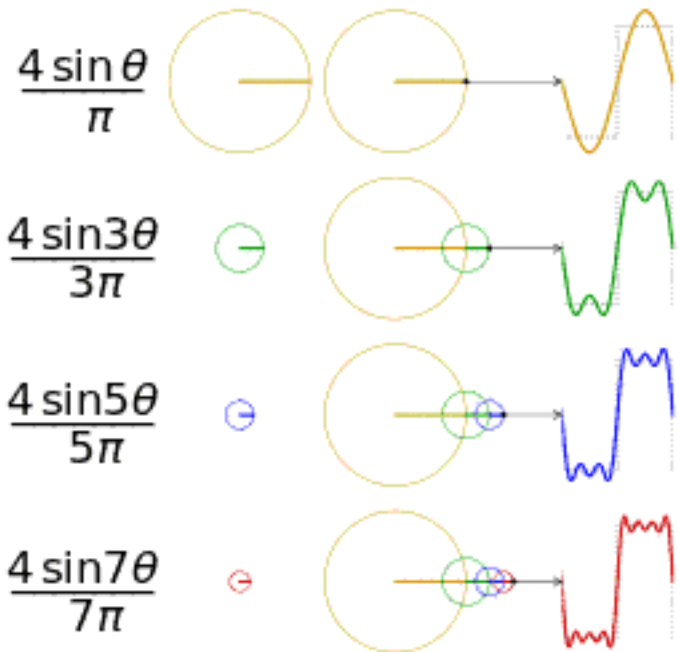
- Periodic function can be represented as a weighted sum of sines and cosines
- Even functions that are not periodic (but whose area under the curve is finite) can be expressed as the integral of sines and/or cosines multiplied by a weight function

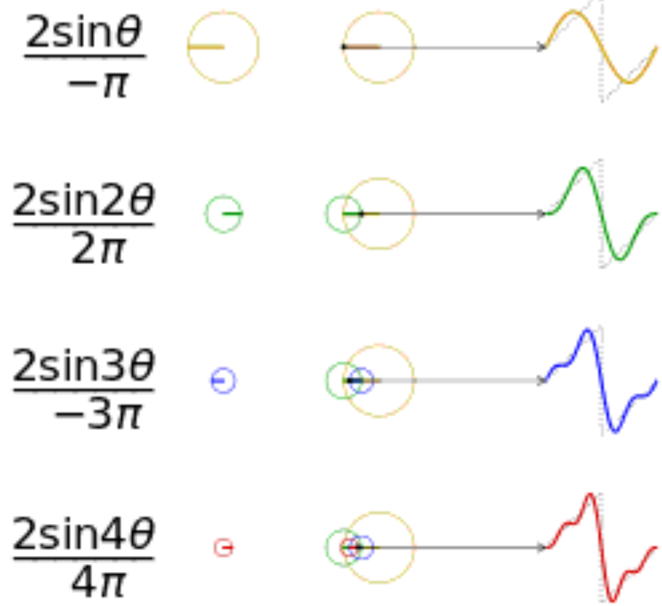


sum =



A sum of sines





https://en.wikipedia.org/wiki/File:Fourier_series_square_wave_circles_animation.gif
https://en.wikipedia.org/wiki/File:Fourier_series_sawtooth_wave_circles_animation.gif

A sum of sines

https://zh.wikipedia.org/wiki/File:Fourier_transform_time_and_frequency_domains_(small).gif

One-Dim Fourier Transform and its Inverse

For a single variable continuous function f(x), the Fourier transform F(u) is defined by:

$$\mathrm{F(u)} = \int_{-\infty}^{\infty} \mathrm{f(x)\,exp(\text{-}j2\pi ux)\,dx} \quad (1)$$

where $j = \sqrt{-1}$

Given F(u), we recover f(x) using the *inverse* Fourier transform:

$$\mathrm{f(x)} = \int_{-\infty}^{\infty} \mathrm{F(u)\,exp(j2\pi ux)\,du} \quad (2)$$

(1) and (2) constitute a **Fourier transform pair**

Two-Dim Fourier Transform and Inverse

- In two dimensions, we have:

$$F(u, v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y) \exp(-j2\pi(ux + vy))\, dx\, dy \qquad (3)$$

$$f(x, y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u, v) \exp(j2\pi(ux + vy))\, du\, dv \qquad (4)$$

Discrete Fourier Transform

- In one dimension,

$$\mathrm{F(u)} = \frac{1}{\mathrm{M}} \sum_{x=0}^{M-1} \mathrm{f(x)} \exp(\text{-j}2\pi \mathrm{ux/M}) \qquad \text{for u} = 0,1,2,\cdots,\mathrm{M}-1. \quad (5)$$

$$\mathrm{f(x)} = \sum_{x=0}^{M-1} \mathrm{F(u)} \exp(\mathrm{j}2\pi \mathrm{ux/M}) \qquad \text{for x} = 0,1,2,\cdots,\mathrm{M}-1. \quad (6)$$

- Note that the location of 1/M does not matter, so long as the product of the two multipliers is 1/M
- Also in the discrete case, the Fourier transform and its inverse always exist

Discrete Fourier Transform

- Consider Euler's formula:

$$\exp^{j\theta} = \cos(\theta i) + j\sin(\theta) \quad (7)$$

- Substituting this expression into (5), and noting

cos(-θ)=cos(θ), we obtain

$$F(u) = \frac{1}{M}\sum_{x=0}^{M-1} f(x)\,[\cos 2\pi ux/M - j\sin 2\pi ux/M], \quad \text{for } u = 0,1,2,\cdots,M-1. \quad (8)$$

- Each term of F depends on all values of f(x), and values of f(x) are multiplied by sines and cosines of different frequencies.
- The domain over which values of F(u) range is called the *frequency domain*, as *u* determines the frequency of the components of the transform.

2-D Discrete Fourier Transform

- Digital images are 2-D discrete functions:

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp(-j2\pi(ux/M + vy/N)),$$
$$\text{for } u = 0,1,2,\cdots,M-1 \text{ and } v = 0,1,2,\cdots,N-1. \quad (9)$$

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)\exp(j2\pi(ux/M + vy/N)),$$
$$\text{for } x = 0,1,2,\cdots,M-1 \text{ and } y = 0,1,2,\cdots,N-1. \quad (10)$$

Frequency Domain Filtering

- Frequency is directly related to rate of change, so frequencies in the Fourier transform may be related to patterns of intensity variations in the image.
- Slowest varying frequency at $u = v = 0$ corresponds to average gray level of the image.
- Low frequencies correspond to slowly varying components in the image- for example, large areas of similar gray levels.
- Higher frequencies correspond to faster gray level changes- such as edges, noise etc.

Procedure for Filtering in the Frequency Domain

1. Multiply the input image by $(-1)^{x+y}$ to centre the transform at (M/2, N/2), which is the centre of the MxN area occupied by the 2D DFT
2. Compute the DFT F(u,v) of the resulting image
3. Multiply F(u,v) by a filter H(u,v)
4. Compute the inverse DFT transform g*(x,y)
5. Obtain the real part g(x,y) of 4
6. Multiply the result by $(-1)^{x+y}$

Example: Notch Filter

- We wish to force the average value of an image to zero. We can achieve this by setting F(0, 0) =0, and then taking its inverse transform.
- So choose the filter function as:

$$\begin{cases} H(u, v) = 0 \text{ if } (u, v) = (M/2, N/2) \\ H(u, v) = 1 \text{ otherwise.} \end{cases}$$

- Called the **notch filter-** constant function with a hole (notch) at the origin.
- A filter that attenuates high frequencies while allowing low frequencies to pass through is called a *lowpass filter*.
- A filter that attenuates low frequencies while allowing high frequencies to pass through is called a *highpass filte*

Convolution Theorem: correspondence between spatial and frequency filtering

- Let $F(u, v)$ and $H(u, v)$ be the Fourier transforms of $f(x, y)$ and $h(x,y)$. Let * be spatial convolution, and multiplication be element-by-element product. Then
  - $f(x, y) * h(x, y)$ and $F(u, v)\, H(u, v)$ constitute a Fourier transform pair , i.e. spatial convolution (LHS) can be obtained by taking the inverse transform of RHS, and conversely, the RHS can be obtained as the forward Fourier transform of LHS.
  - Analogously, convolution in the frequency domain reduces to multiplication in the spatial domain, and vice versa.
- Using this theorem, we can also show that filters in the spatial and frequency domains constitute a Fourier transform pair.

Exploiting the correspondence

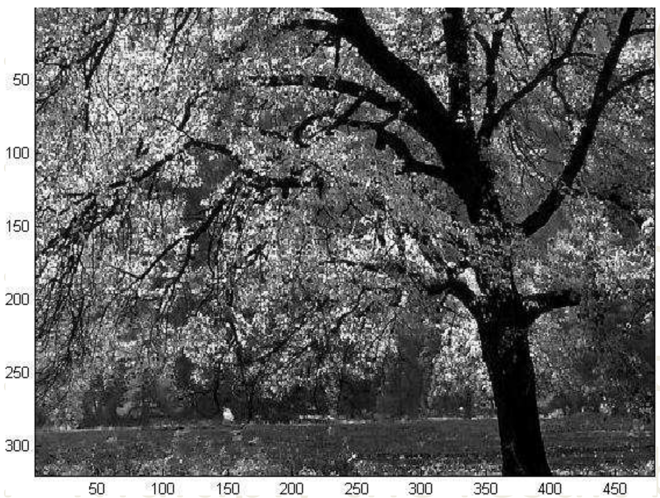
- If filters in the spatial and frequency domains are of the same size, then filtering is more efficient computationally in frequency domain.
- However, spatial filters tend to be smaller in size.
- Filtering is also more intuitive in frequency domain- so design it there.
- Then, take the inverse transform, and use the resulting filter as a guide to design smaller filters in the spatial domain.

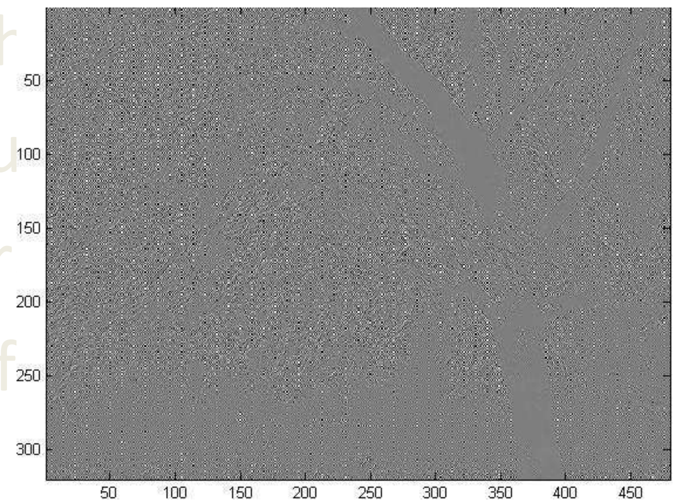
Example of Smoothing an Image

- In spatial domain, we just convolve the image with a Gaussian kernel to smooth it
- In frequency domain, we can multiply the image by a filter achieve the same effect

Example of Smoothing an Image

1. Multiply the input image by $(-1)^{x+y}$ to center the transform

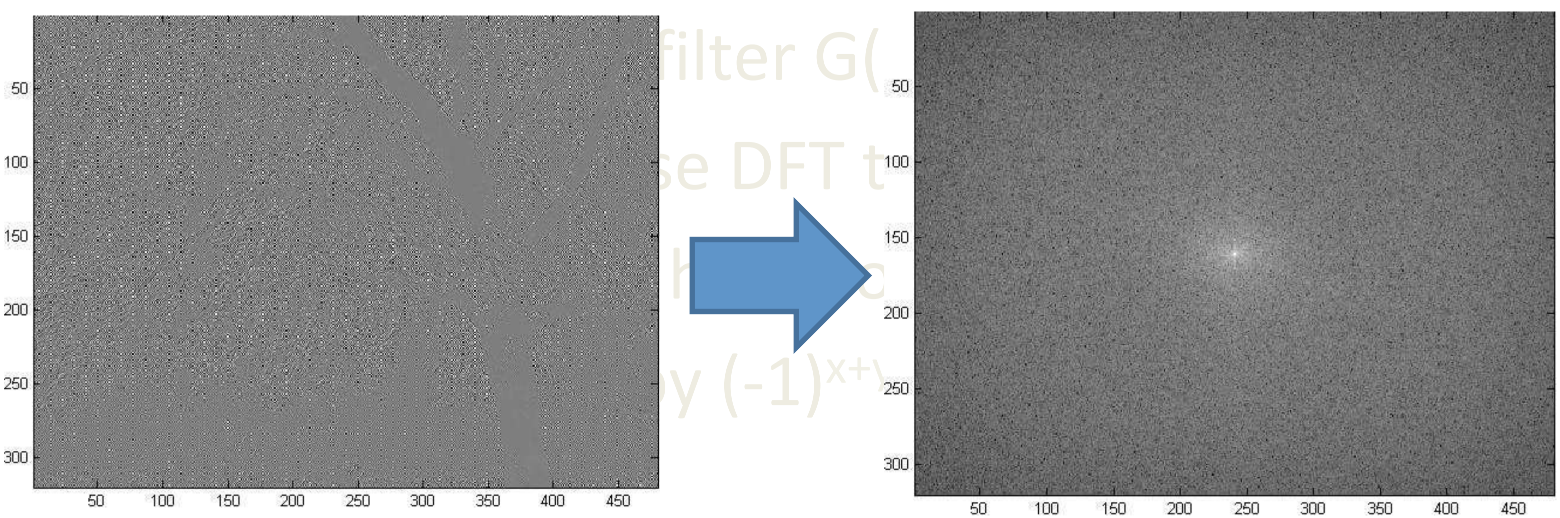




Example of Smoothing an Image

1. Multiply the input image by $(-1)^{x+y}$ to center the transform

2. Compute the DFT F(u,v) of the resulting image



3. filter G(

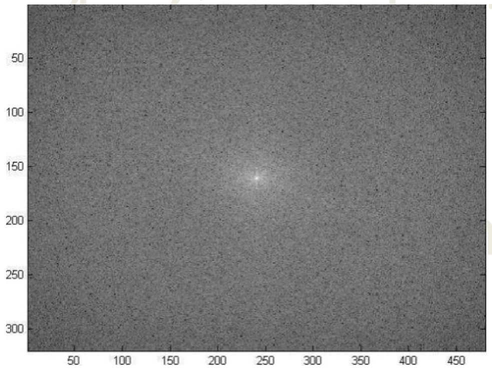
4. se DFT t

5. h o

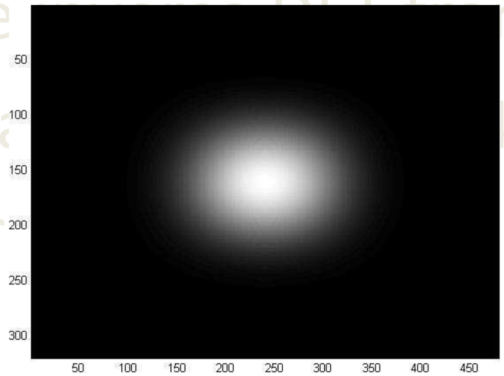
6. y $(-1)^{x+y}$

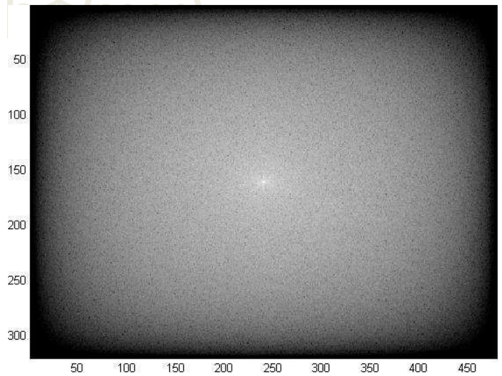
Example of Smoothing an Image

1. Multiply the input image by $(-1)^{x+y}$ to center the transform
2. Compute the DFT F(u,v) of the resulting image
3. Multiply F(u,v) by a filter G(u,v)



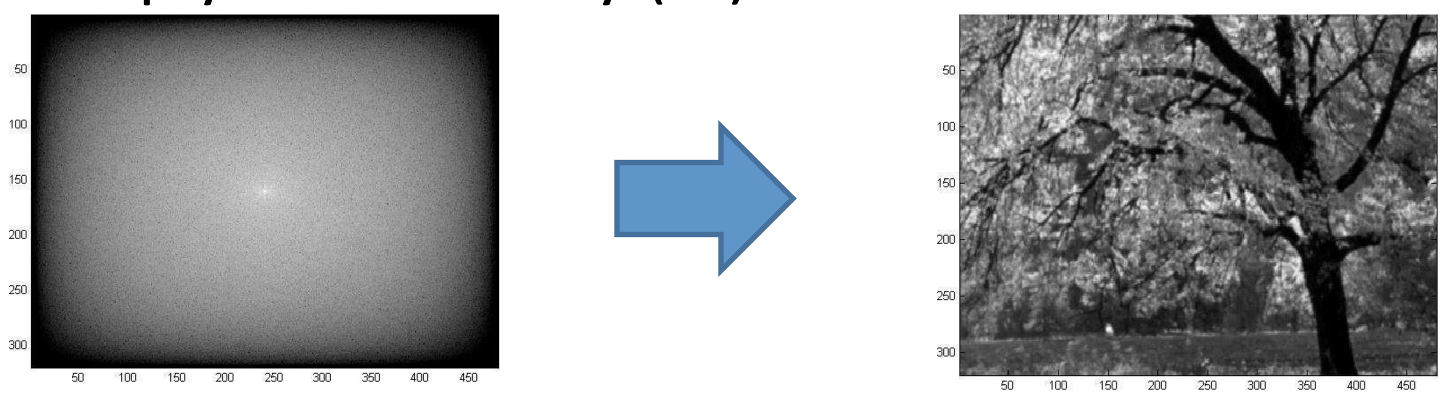
X





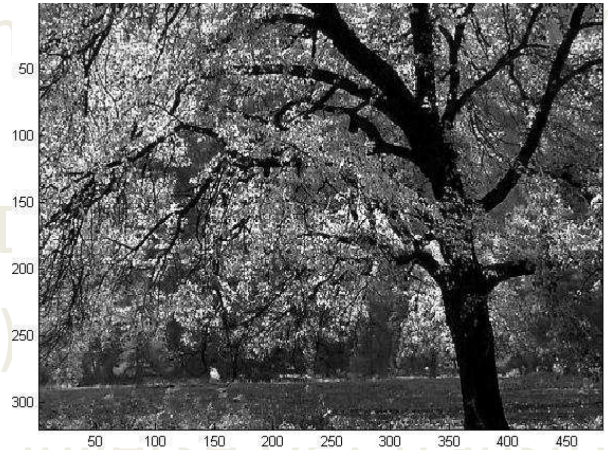
Example of Smoothing an Image

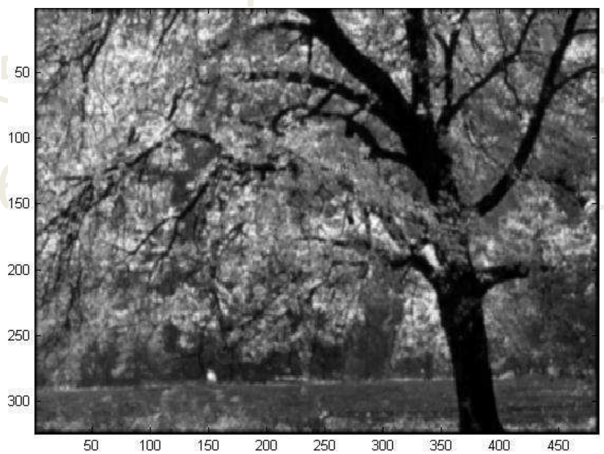
1. Multiply the input image by $(-1)^{x+y}$ to center the transform
2. Compute the DFT F(u,v) of the resulting image
3. Multiply F(u,v) by a filter G(u,v)
4. Compute the inverse DFT transform h*(x,y)
5. Obtain the real part h(x,y) of 4
6. Multiply the result by $(-1)^{x+y}$



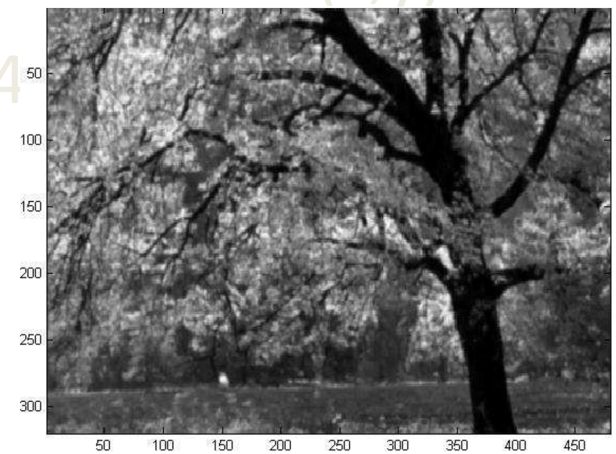
Example of Smoothing an Image

1. Multiply the in … to center the transform
2. Compute the D … ulting image
3. Multiply F(u,v)
4. Computer the inverse DFT transform h*(x,y)
5. … part h(x,y) of 4
6. … ult by $(-1)^{x+y}$





$f(x, y) * g(x, y)$



$F(u, v)G(u, v)$

Gaussian Filter

- Gaussian filters are important because their shapes are easy to specify, and both the forward and inverse Fourier transforms of a Gaussian function are real Gaussian functions.
- Let H(u) be a one dimensional Gaussian filter specified by:

$$\mathrm{H(u)} = \mathrm{A}\exp^{-\frac{u^2}{2\sigma^2}}$$

where σ is the standard deviation of the Gaussian curve.

- The corresponding filter in the spatial domain is

$$\mathrm{h(x)} = \sqrt{2\pi\sigma}\,\mathrm{A}\exp^{-2\pi^2\sigma^2x^2}$$

- This is usually a lowpass filter.

DoG Filter

- Difference of Gaussians may be used to construct highpass filters:

$$\mathrm{H(u)} = \mathrm{A}\exp^{-\frac{u^2}{2\sigma_1^2}} - \mathrm{B}\exp^{-\frac{u^2}{2\sigma_2^2}}$$

with $A \geq B$ and $\delta_1 > \delta_2$.

- The corresponding filter in the spatial domain is

$$\mathrm{h(x)} = \sqrt{2\pi\sigma_1}\,\mathrm{A}\exp^{-2\pi^2\sigma_1^2x^2} - \sqrt{2\pi\sigma_2}\,\mathrm{B}\exp^{-2\pi^2\sigma_2^2x^2}$$

Image Pyramids

- An image pyramid is a collection of decreasing resolution images arranged in the shape of a pyramid.

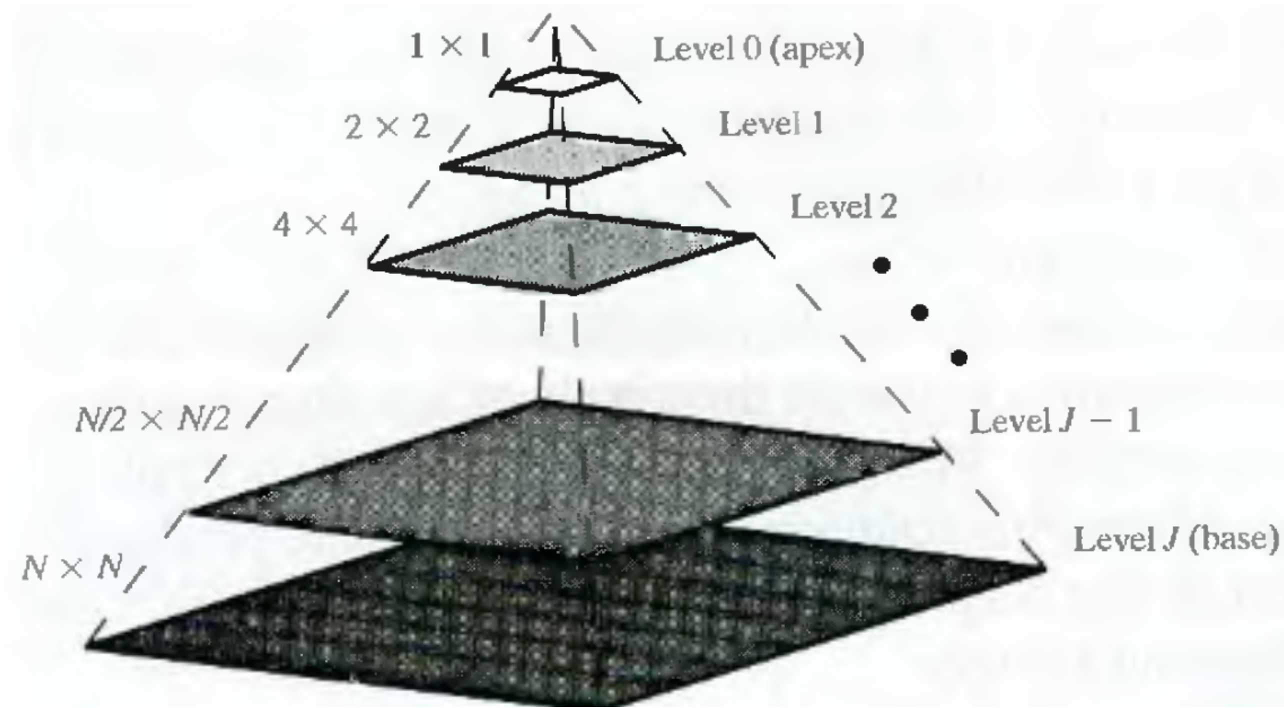
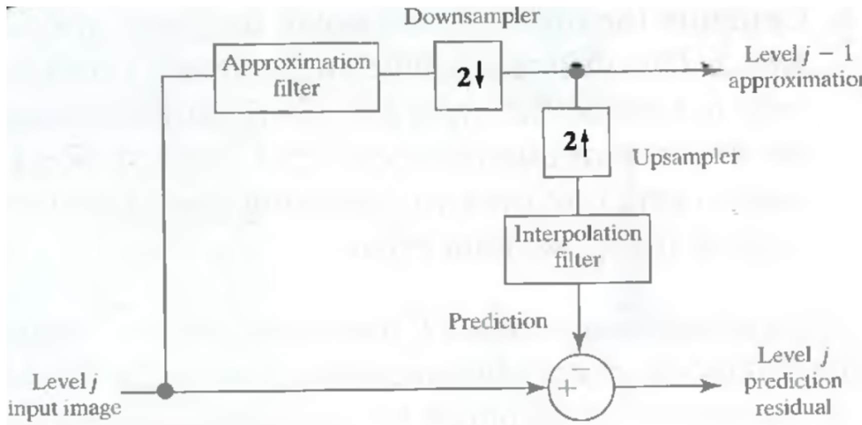


Image Pyramids

System block diagram for creating image pyramids

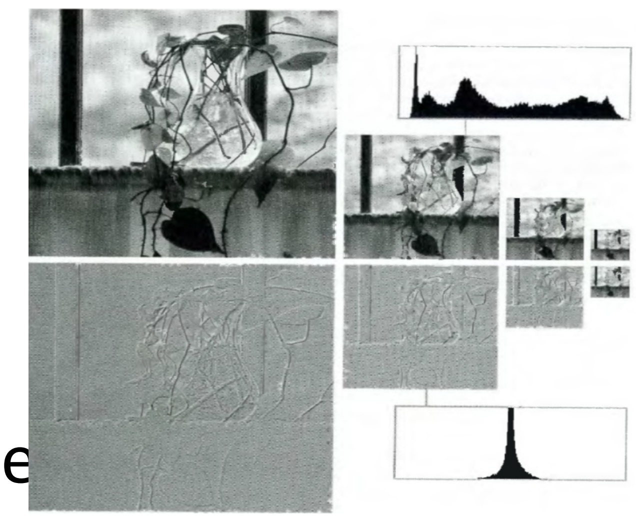


1. Compute a reduced-resolution approximation of the input image by filtering and downsampling (mean, Gaussian, subsampling)
2. Upsample the output of step 1 and filter the result (possibly with interpolation)
3. Compute the difference between the prediction of step 2 and the input to step 1

Repeating, produce approximation and prediction residual pyramids

Image Pyramids

Two image pyramids and their statistics (Gaussian approx pyramid, Laplacian prediction residual pyramid)



To recreate image

- Upsample and filter the lowest resolution approximation image
- Add the 1-level higher Laplacian's prediction residual

References and Acknowledgement

- *Gonzalez* and Woods, 2002, *Chapter 4.1-4.4, 7.1*
- *Szeliski Chapter 3.1-3.5*
- Some material, including images and tables, were drawn from the textbook, *Digital Image Processing* by Gonzalez and Woods, and P.C. Rossin's presentation.