南京大学
NANJING UNIVERSITY

计算机视觉表征与识别
Chapter 3: Frequency Domain and Sampling

王利民

媒体计算课题组

http://mcg.nju.edu.cn/

Image filtering

- Compute a function of the **local neighborhood** at each pixel in the image
  - Function specified by a “filter” or mask saying how to combine values from neighbors.

- Uses of filtering:
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (template matching)

Correlation filtering

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$

This is called cross-correlation, denoted $G = H \otimes F$

Filtering an image: replace each pixel with a linear combination of its neighbors.

The filter “kernel” or “mask” *H*[*u,v*] is the prescription for the weights in the linear combination.

Averaging filter

- What values belong in the kernel *H* for the moving average example?

$F[x, y]$ $\otimes$ $H[u, v]$ $G[x, y]$

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

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

"box filter"

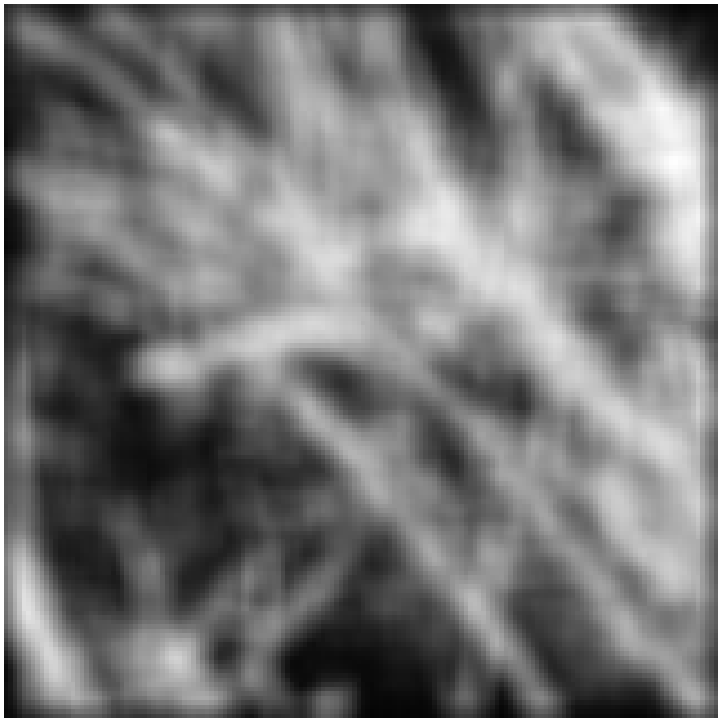
	0	20	40	60	60				

$$G = H \otimes F$$

Smoothing by averaging

depicts box filter:
white = high value, black = low value

original



filtered

What if the filter size was 5 x 5 instead of 3 x 3?

Gaussian filter

- What if we want nearest neighboring pixels to have the most influence on the output?

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

$F[x, y]$

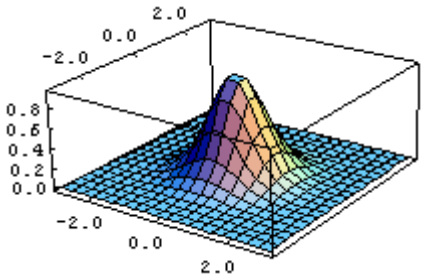
$$\frac{1}{16}$$

1	2	1
2	4	2
1	2	1

$H[u, v]$

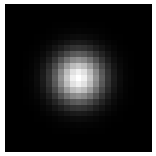
This kernel is an approximation of a 2d Gaussian function:

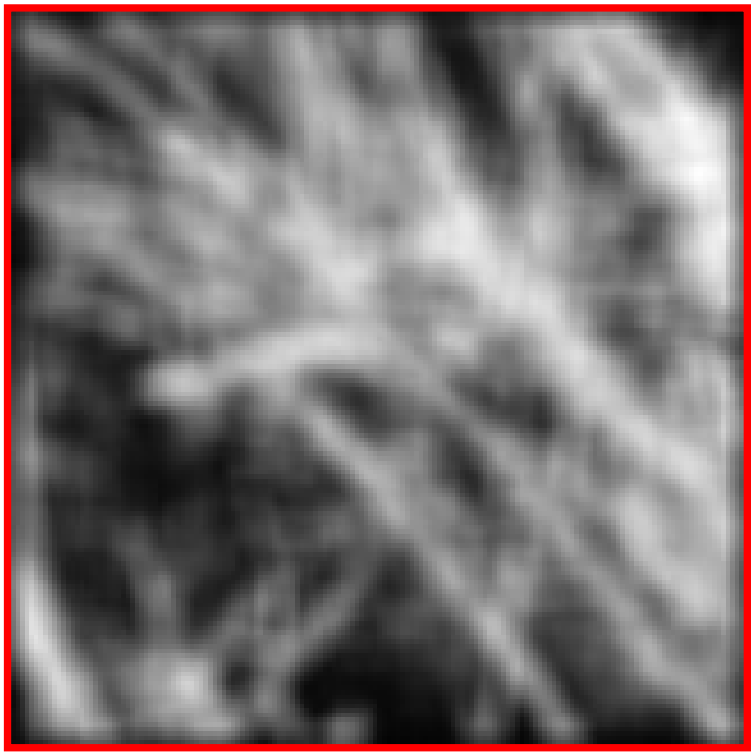
$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



- Removes high-frequency components from the image (“low-pass filter”).

Smoothing with a Gaussian





Convolution vs correlation

Definition of discrete 2D convolution:

notice the flip

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j)I(x - i, y - j)$$

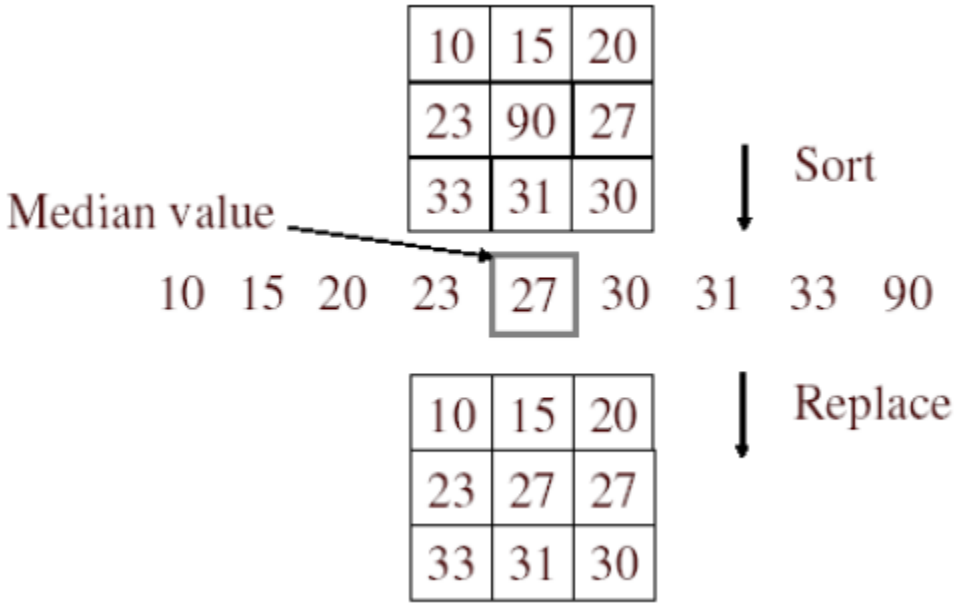
Definition of discrete 2D correlation:

notice the lack of a flip

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j)I(x + i, y + j)$$

- Most of the time won't matter, because our kernels will be symmetric.

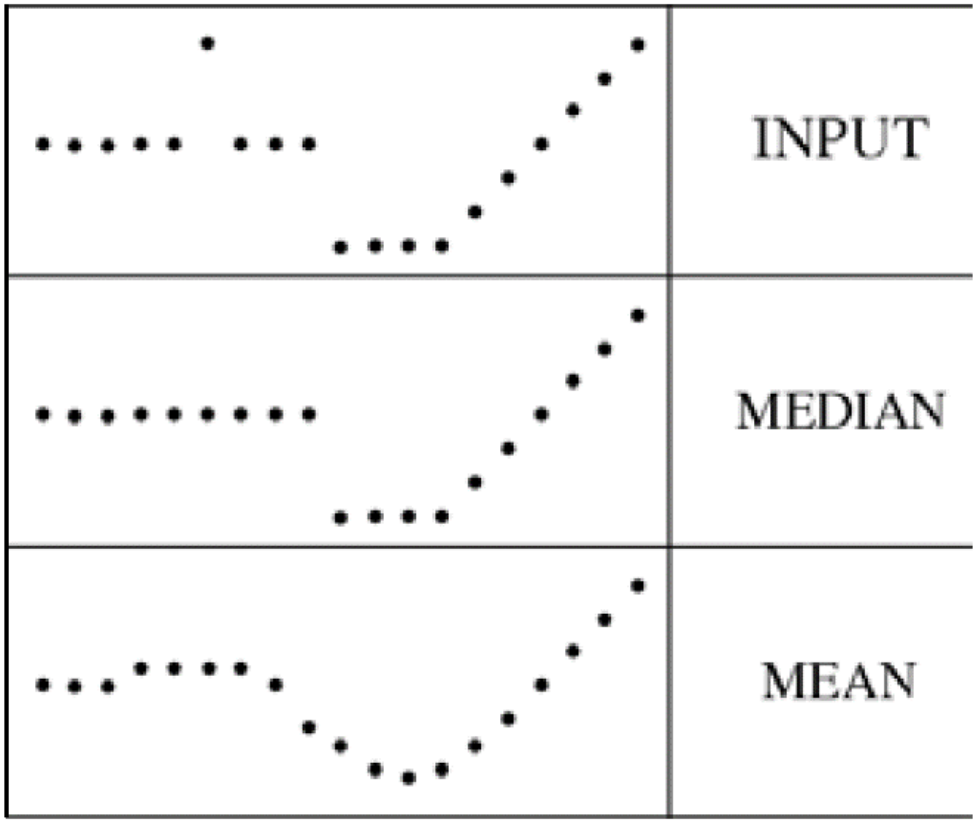
Median filter



- No new pixel values introduced
- Removes spikes: good for impulse, salt & pepper noise
- Non-linear filter

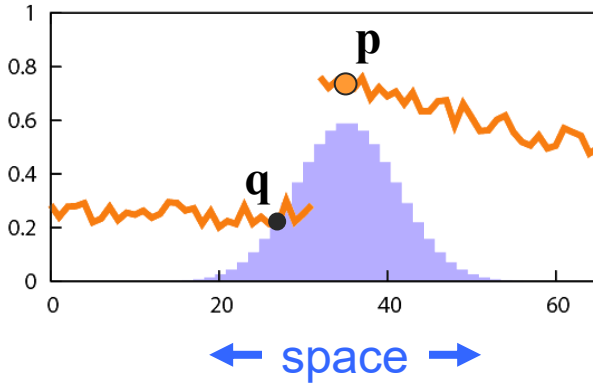
Median filter

- Median filter is edge preserving



Definition

Gaussian blur



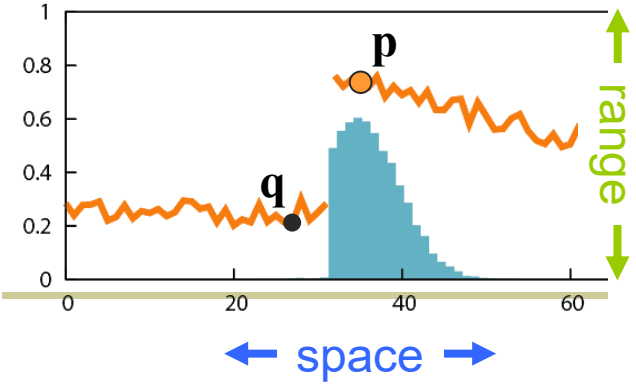
$$I_{\mathbf{p}}^{\mathrm{b}} = \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)\, I_{\mathbf{q}}$$

space

- only spatial distance, intensity ignored

Bilateral filter

[Aurich 95, Smith 97, Tomasi 98]

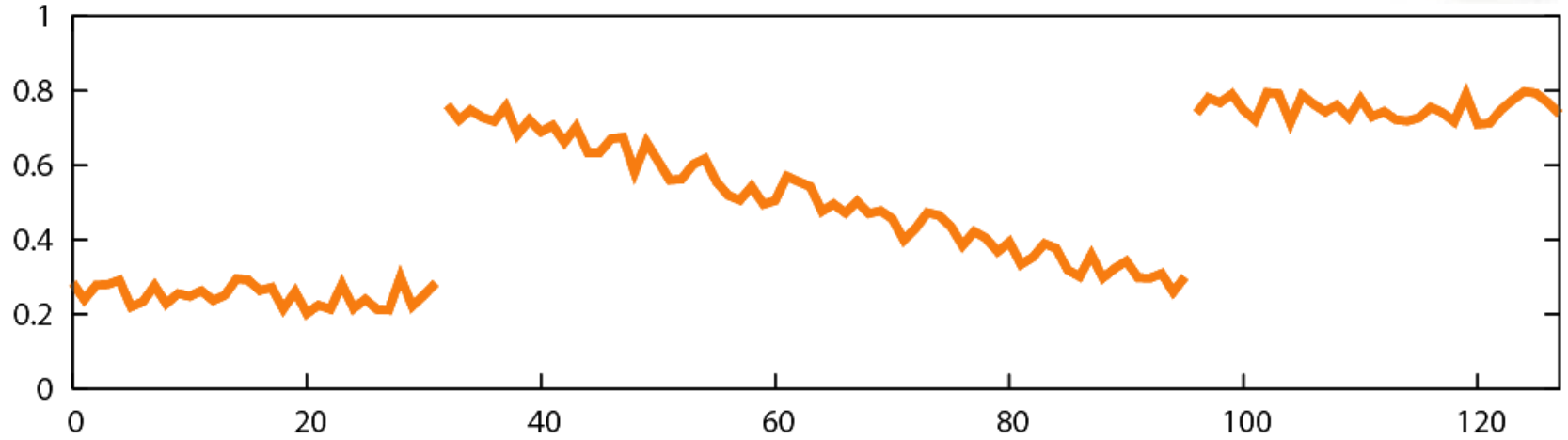


$$I_{\mathbf{p}}^{\mathrm{bf}} = \frac{1}{W_{\mathbf{p}}^{\mathrm{bf}}} \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_{\mathrm{r}}}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)\, I_{\mathbf{q}}$$

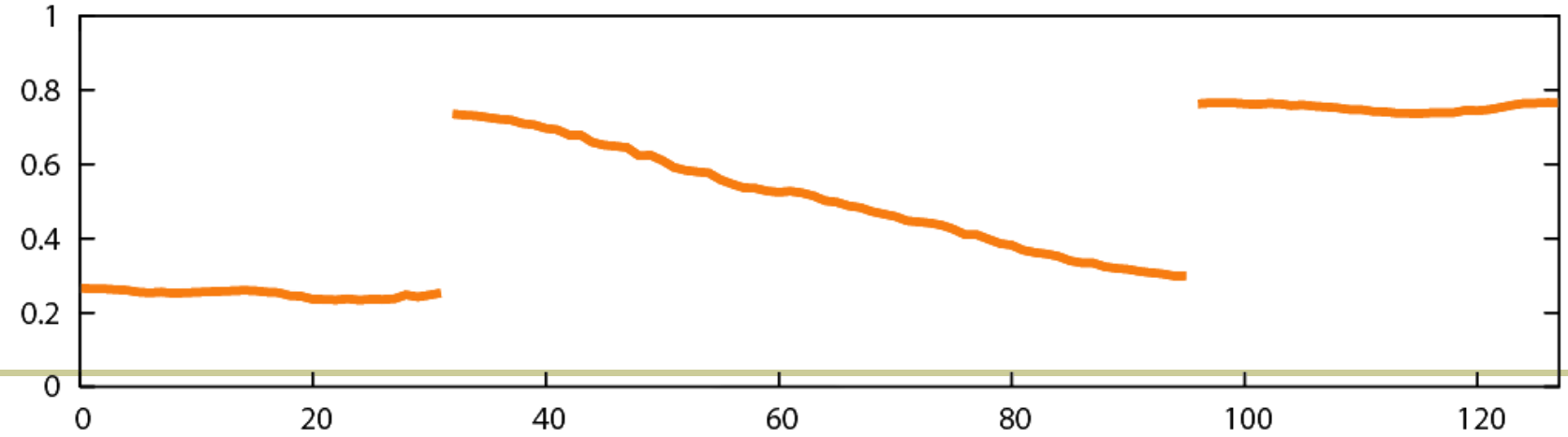
normalization　space　range

- spatial and range distances
  - weights sum to 1

Bilateral Filter on 1D Signal



BF



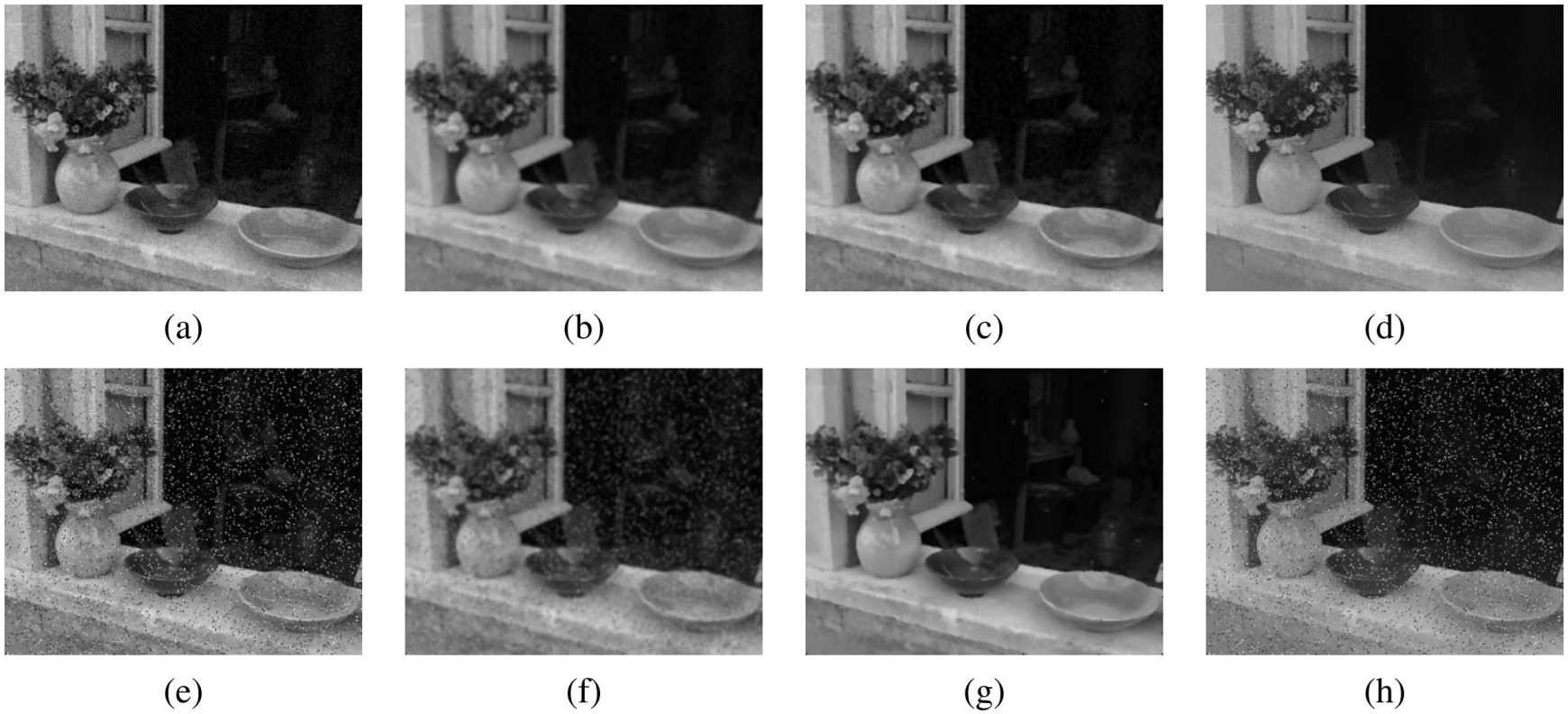


Figure 3.18 Median and bilateral filtering: (a) original image with Gaussian noise; (b) Gaussian filtered; (c) median filtered; (d) bilaterally filtered; (e) original image with shot noise; (f) Gaussian filtered; (g) median filtered; (h) bilaterally filtered. Note that the bilateral filter fails to remove the shot noise because the noisy pixels are too different from their neighbors.

Today's Class

- Frequency domain and Fourier transform
- 1D Discrete Fourier Transform
- 2D Image Fourier Transform
- Sampling
- Hybrid Image

Three views of filtering

- Image filters in spatial domain
  - Filter is a mathematical operation on values of each patch
  - Smoothing, sharpening, measuring texture
- Image filters in the frequency domain
  - Filtering is a way to modify the frequencies of images
  - Denoising, sampling, image compression
- Templates and Image Pyramids
  - Filtering is a way to match a template to the image
  - Detection, coarse-to-fine registration

Jean Baptiste Joseph Fourier (1768-1830)

had crazy idea (1807):

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

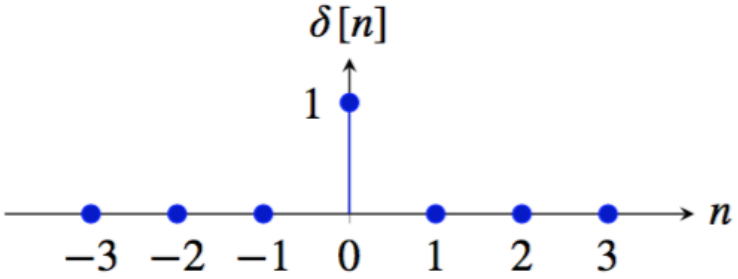
- Don't believe it?
  - Neither did Lagrange, Laplace, Poisson and other big wigs
  - Not translated into English until 1878!
- But it's (mostly) true!
  - called Fourier Series
  - there are some subtle restrictions

...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.

Slides: Efros

Important signals

The impulse



The result of convolving a signal *g*[n] with the impulse signal is the same signal:

$$f[n] = \delta \circ g = \sum_k \delta[n-k]\, g[k] = g[n]$$

Convolving a signal *f* with a translated impulse δ [*n* − *n*0] results in a translated signal:

$$f[n-n_0] = \delta[n-n_0] \circ f[n]$$

Important signals

Cosine and sine waves

$$s(t) = A\sin(w\,t - \theta)$$

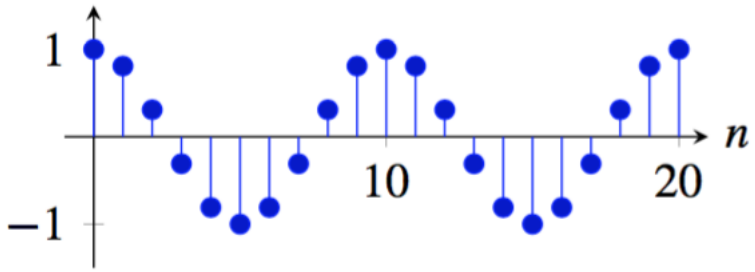
$$s_k[n] = \sin\left(\frac{2\pi}{N}kn\right) \qquad c_k[n] = \cos\left(\frac{2\pi}{N}kn\right)$$

$k \in [1, N/2]$ denotes the number of wave cycles that will occur within the region of support

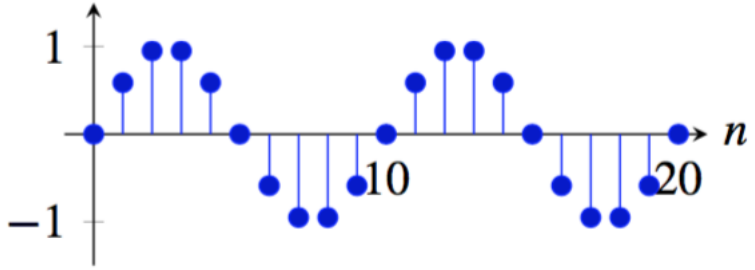
Important signals

Cosine and sine waves

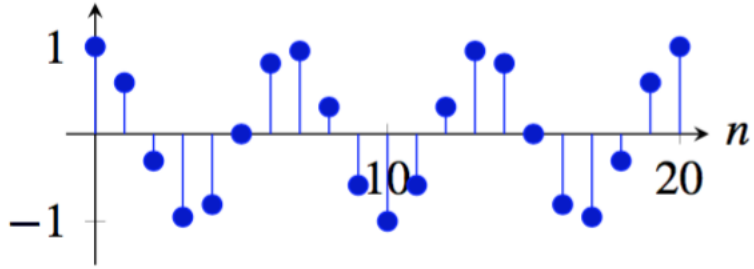
$\cos\left(\frac{2\pi}{N}kn\right)$ k=2



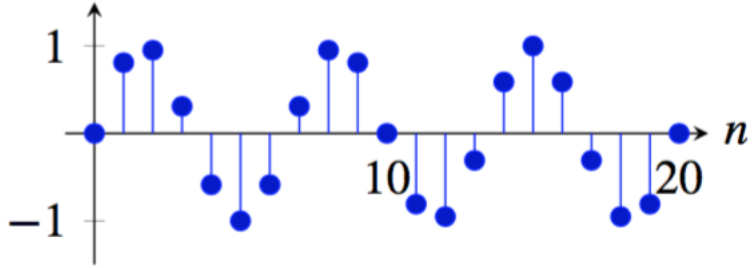
$\sin\left(\frac{2\pi}{N}kn\right)$ k=2



$\cos\left(\frac{2\pi}{N}kn\right)$ k=3



$\sin\left(\frac{2\pi}{N}kn\right)$ k=3



Important signals

Complex exponential

$$s(t) = A\exp(jwt)$$

In discrete time (setting $A = 1$), we can write:

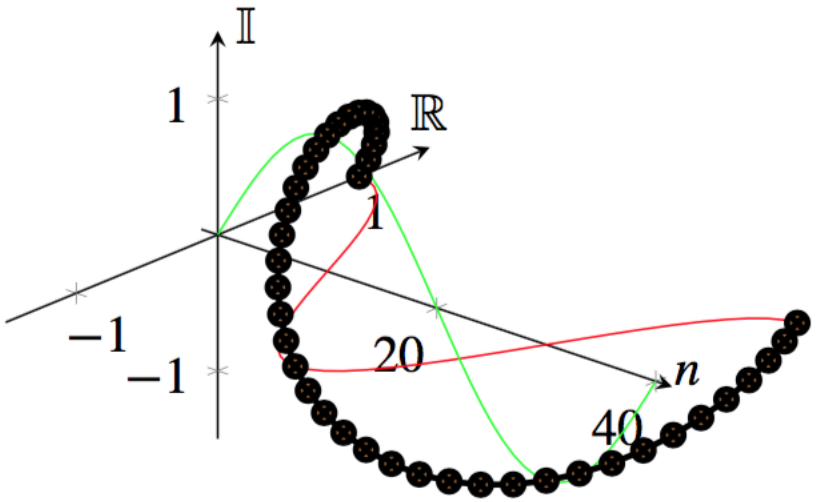
$$e_k[n] = \exp\left(j\frac{2\pi}{N}kn\right) = \cos\left(\frac{2\pi}{N}kn\right) + j\sin\left(\frac{2\pi}{N}kn\right)$$

And in 2D, the complex exponential wave is:

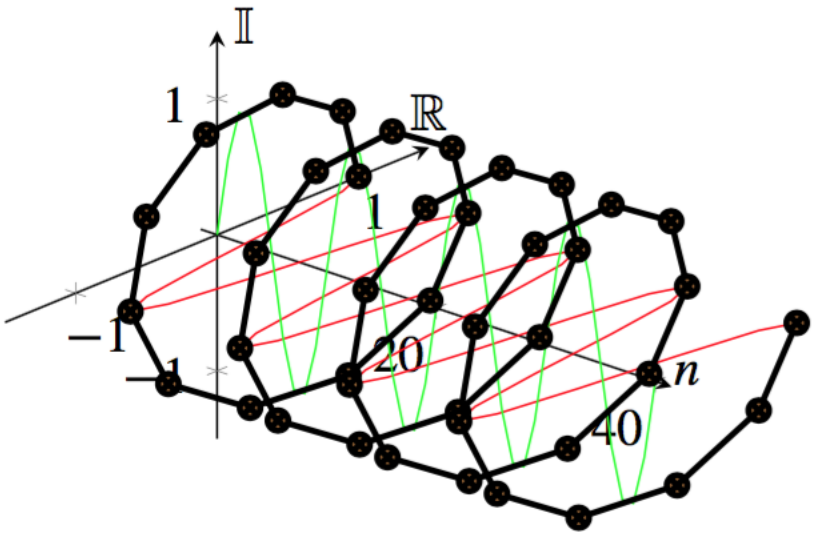
$$e_{u,v}[n,m] = \exp\left(2\pi j\left(\frac{un}{N} + \frac{vm}{M}\right)\right)$$

Important signals

Complex exponential



$N = 40, k = 1$



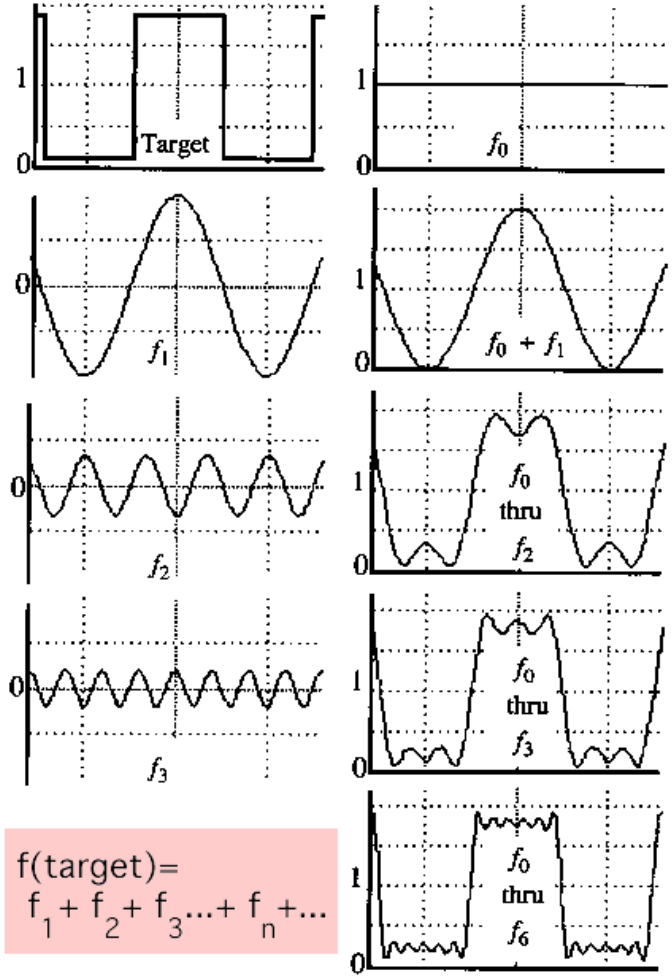
$N = 40, k = 4$

A simple idea: a sum of sines

Our building block:

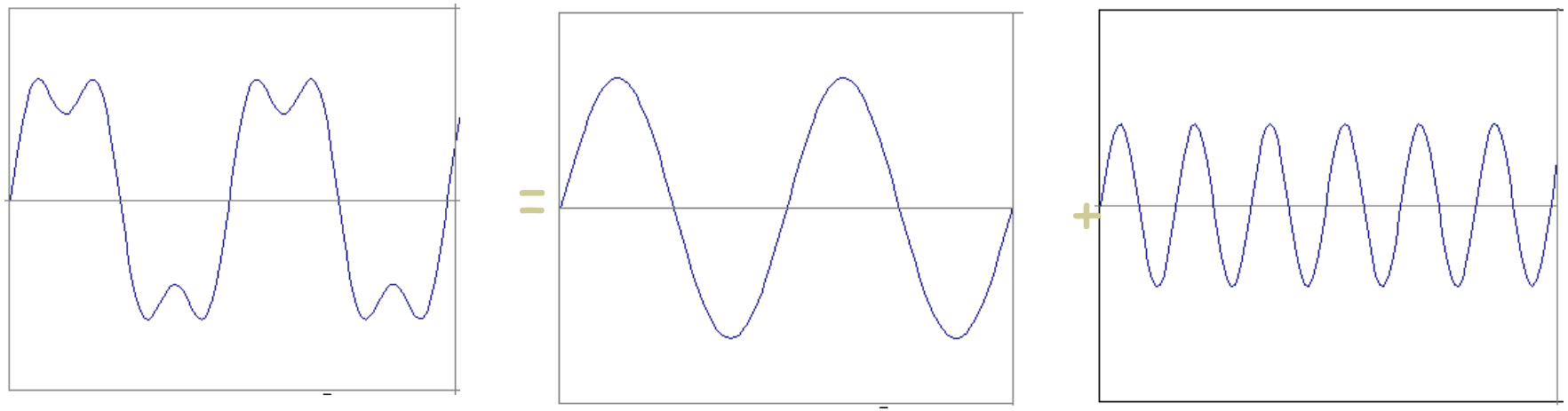
$$A\sin(\omega x + \phi)$$

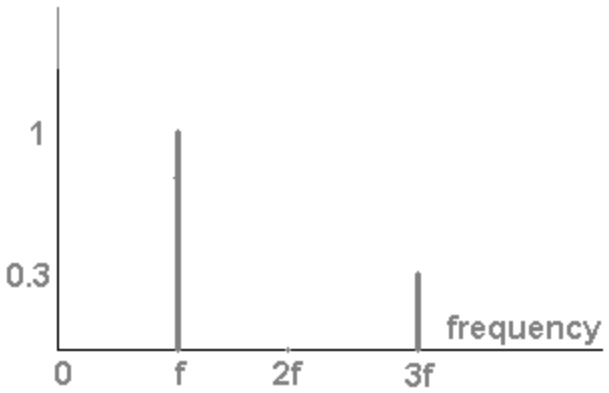
Add enough of them to get any signal *f(x)* you want!



Frequency Spectra

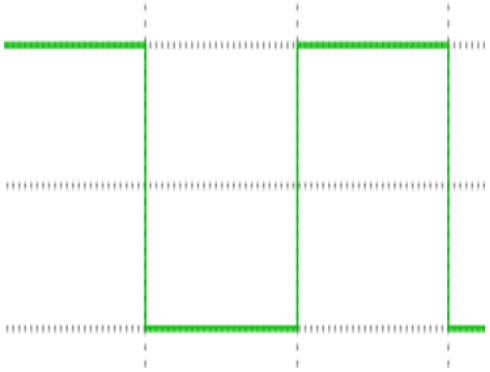
- example : $g(t) = \sin(2\pi f\, t) + (1/3)\sin(2\pi(3f)\, t)$



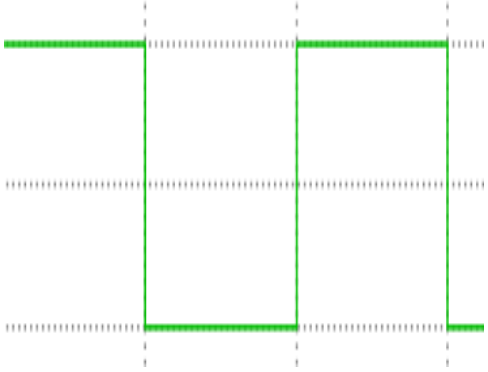


Slides: Efros

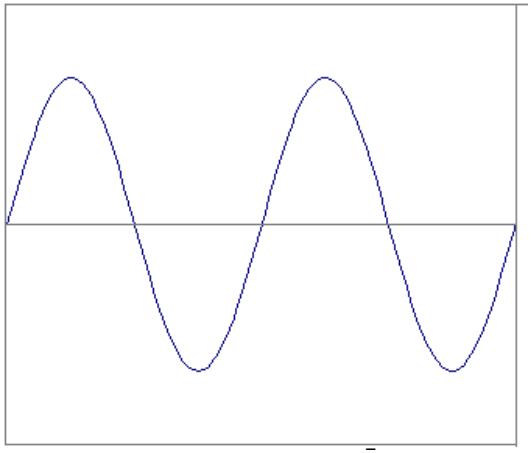
Frequency Spectra



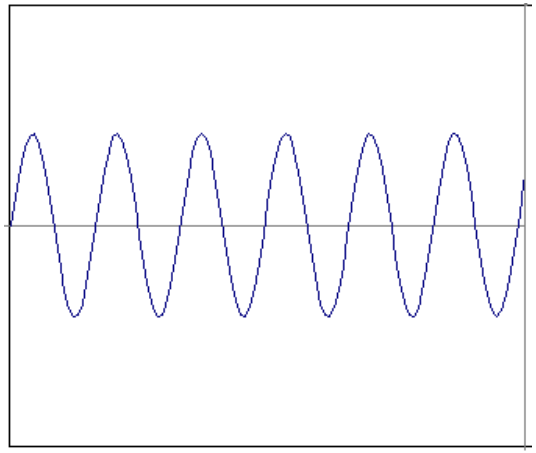
Frequency Spectra



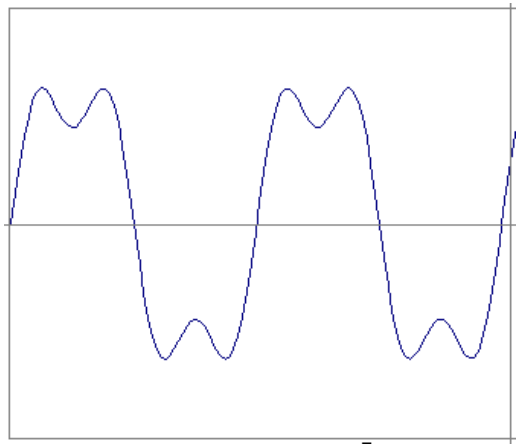
=



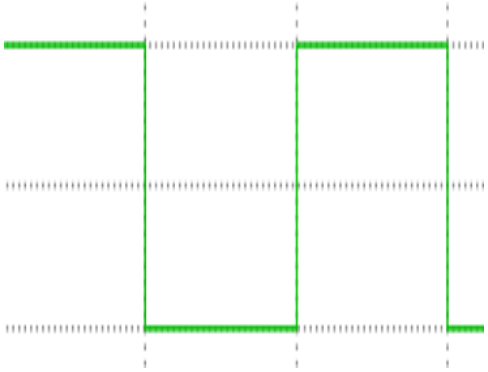
+



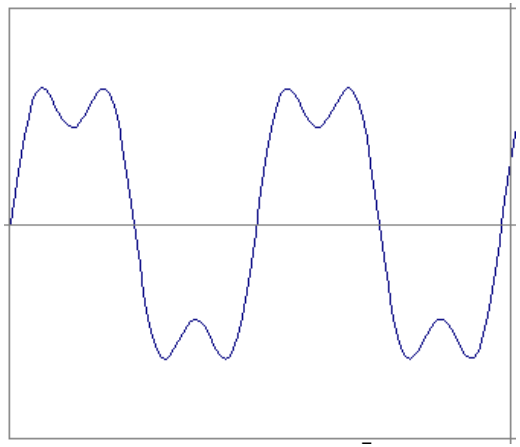
=



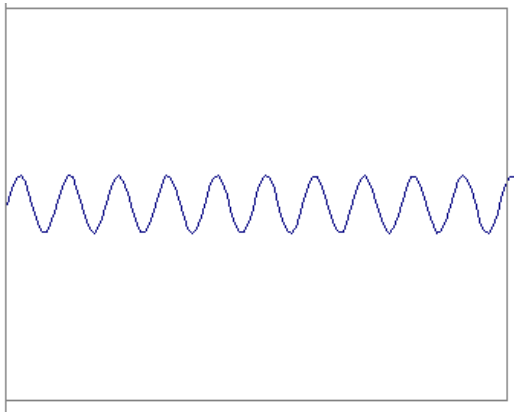
Frequency Spectra



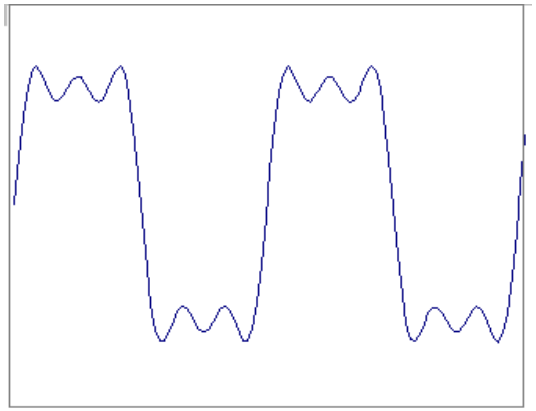
=



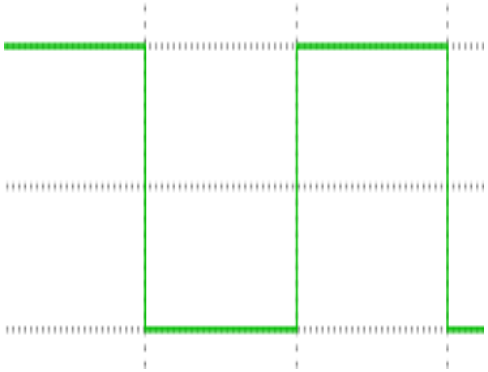
+



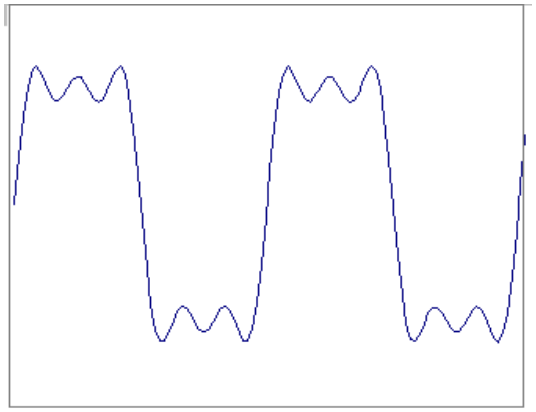
=



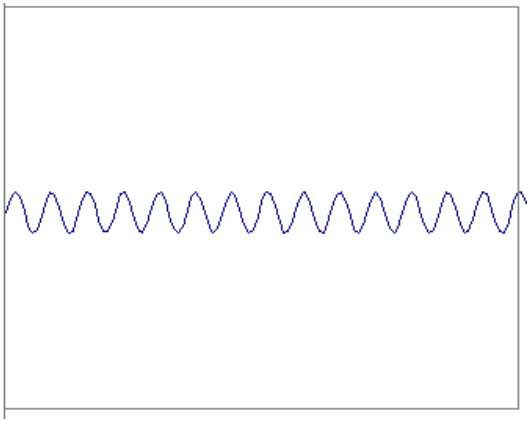
Frequency Spectra



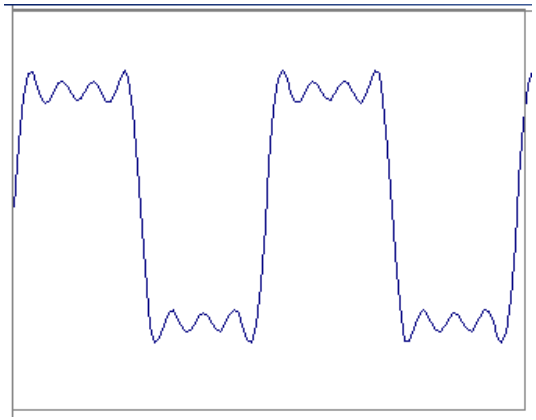
=



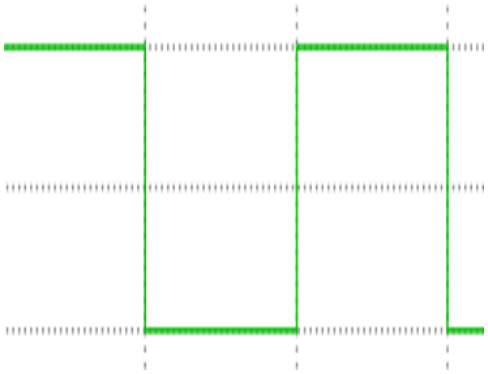
+



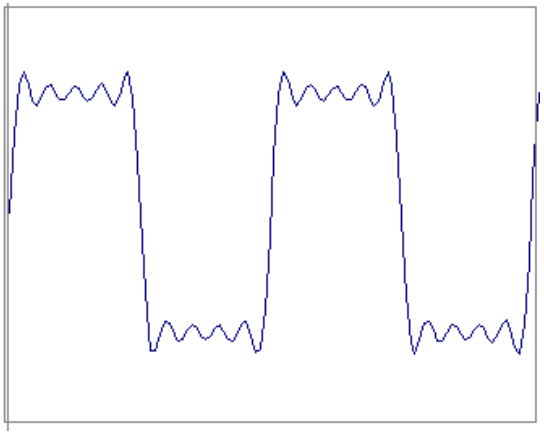
=



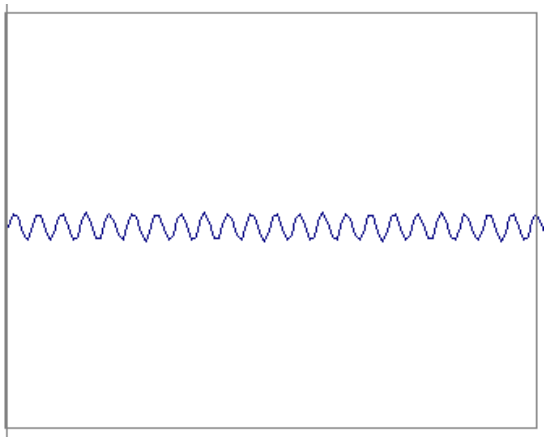
Frequency Spectra



=

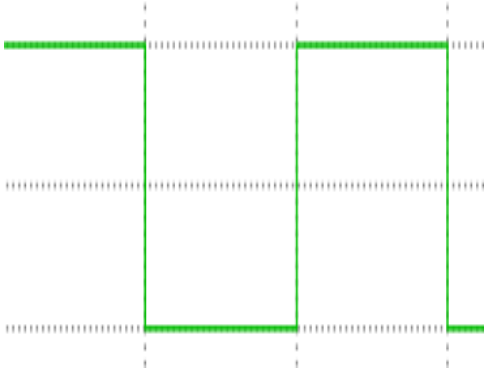


+

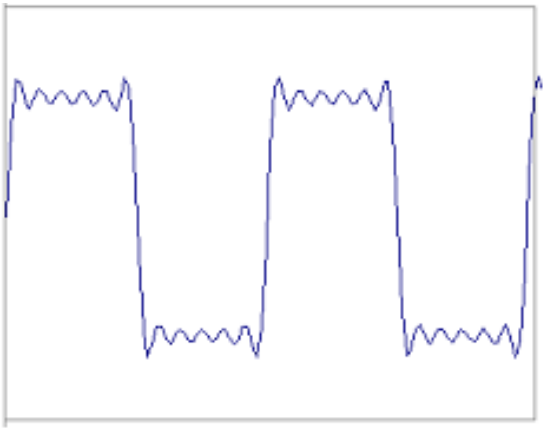


=

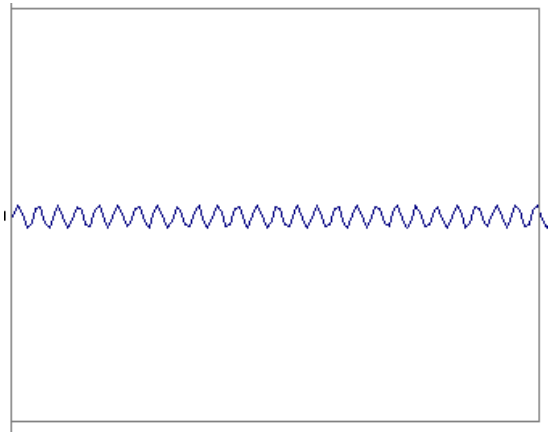
Frequency Spectra



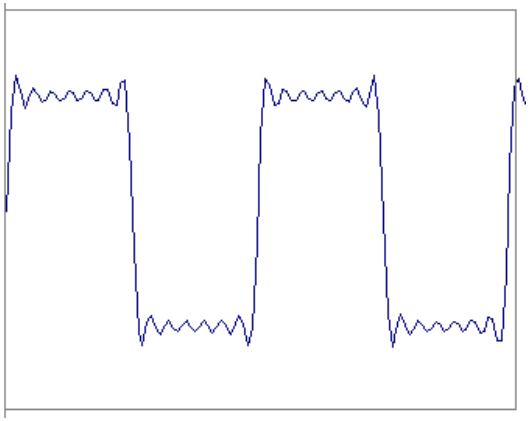
=



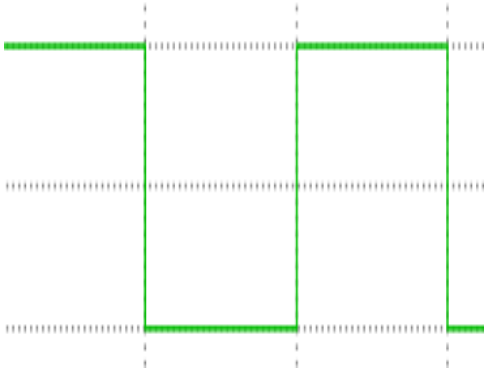
+



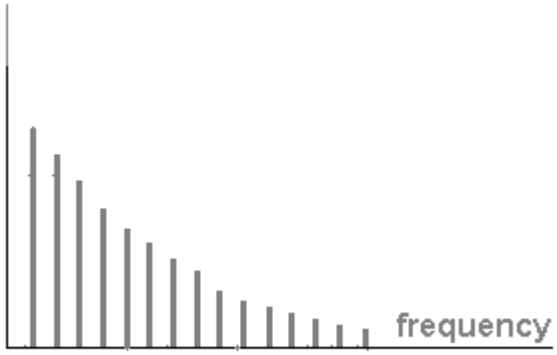
=



Frequency Spectra



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



Linear image transformations

- In analyzing images, it is often useful to make a change of basis.

Transformed image

$$\vec{F} = U\vec{f}$$

Vectorized image

Fourier transform, or
Wavelet transform, or
Steerable pyramid transform

$$= \begin{vmatrix} U \end{vmatrix}$$

Self-inverting transforms

$$\vec{F} = U\vec{f} \quad \Longleftrightarrow \quad \vec{f} = U^{-1}\vec{F}$$

Same basis functions are used for the inverse transform

$$\vec{f} = U^{-1}\vec{F}$$

$$= U^{+}\vec{F}$$

U transpose and complex conjugate

Today's Class

- Frequency domain and Fourier transform
- **1D Discrete Fourier Transform**
- 2D Image Fourier Transform
- Sampling
- Hybrid Image

The Discrete Fourier transform

Discrete Fourier Transform (DFT) transforms a signal $f[n]$ into $F[u]$ as:

$$F[u] = \sum_{n=0}^{N-1} f[n] \exp\left(-2\pi j \frac{un}{N}\right)$$

The inverse of the DFT is:

$$f[n] = \frac{1}{N} \sum_{u=0}^{N-1} F[u] \exp\left(2\pi j \frac{un}{N}\right)$$

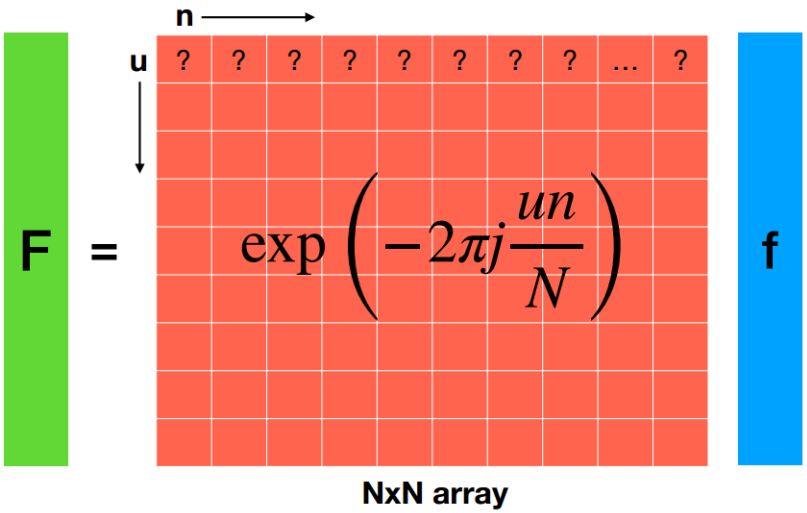
The signal $f[n]$ is a weighted linear combination of complex exponentials with weights $F[u]$

The Discrete Fourier transform

Discrete Fourier Transform (DFT) transforms a signal $f[n]$ into $F[u]$ as:

$$F[u] = \sum_{n=0}^{N-1} f[n] \exp\left(-2\pi j \frac{un}{N}\right)$$

Discrete Fourier Transform (DFT) is a linear operator. Therefore, we can write:



Lets visualize the transform coefficients

69

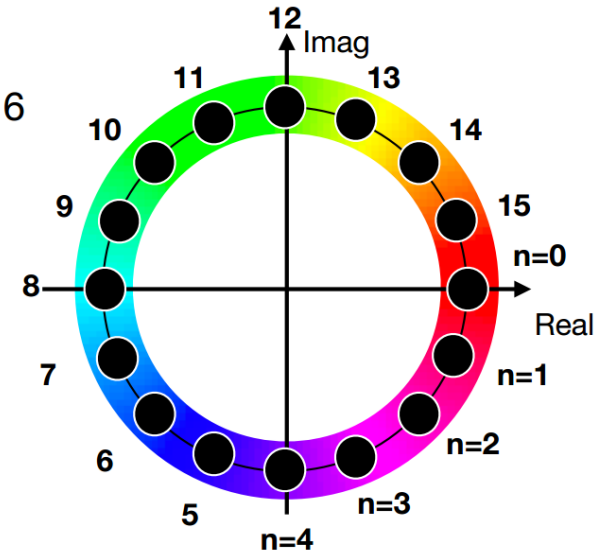
Visualizing the Fourier transform

$$F[u] = \sum_{n=0}^{N-1} f[n] \exp\left(-2\pi j \frac{un}{N}\right)$$

$$\exp(\alpha\, j) = \cos(\alpha) + j\, \sin(\alpha)$$

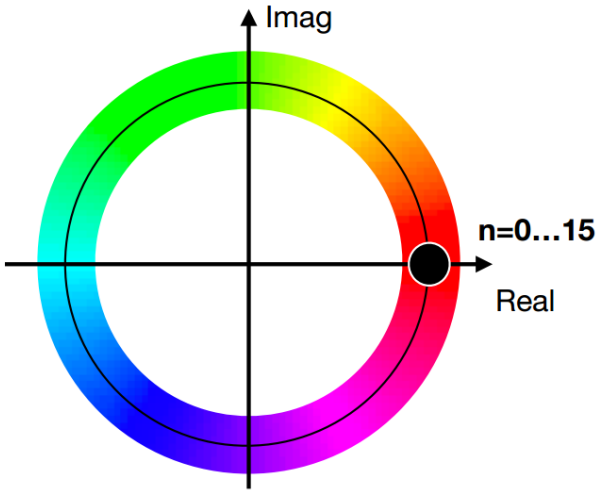
$$\cos\left(2\pi \frac{un}{N}\right) - j\ \sin\left(2\pi \frac{un}{N}\right)$$

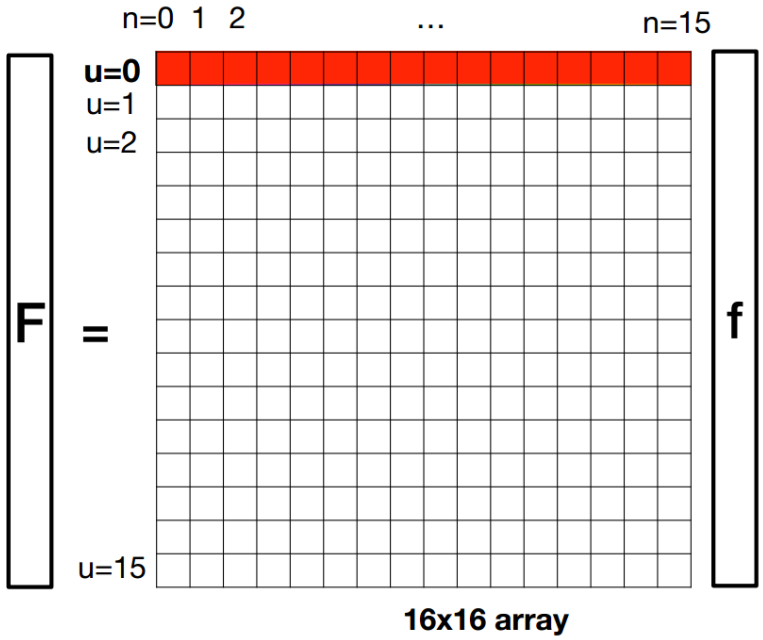
For:
u=1
N=16



Visualizing the transform coefficients

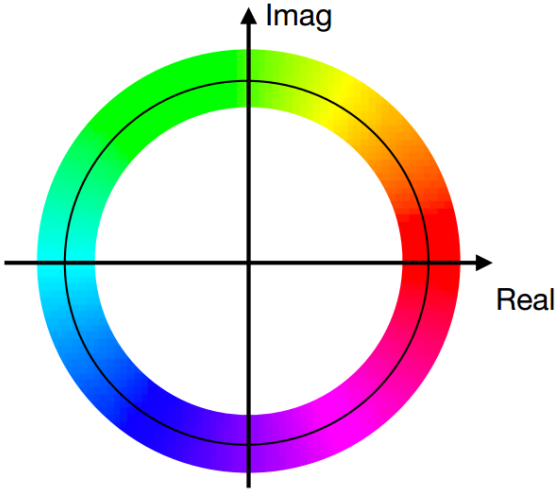
$$\exp\left(-2\pi j\frac{un}{N}\right)$$ For N=16

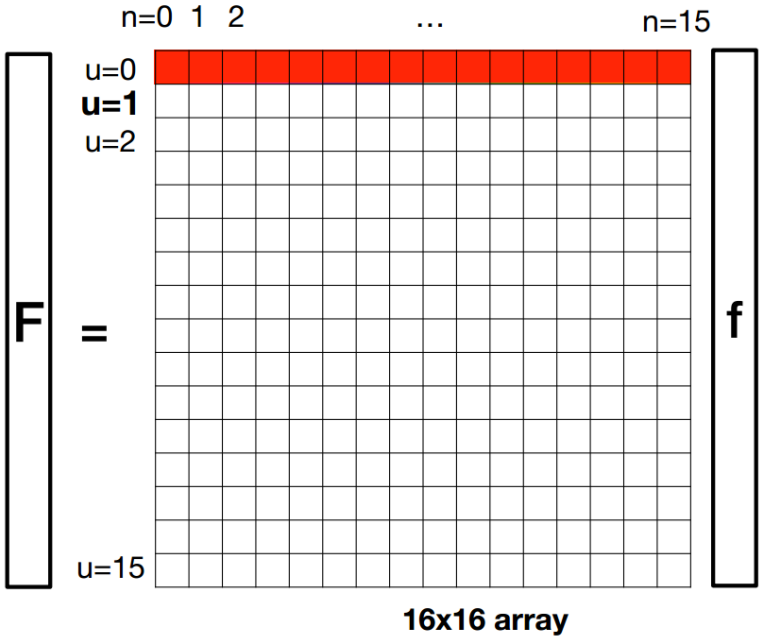




Visualizing the transform coefficients

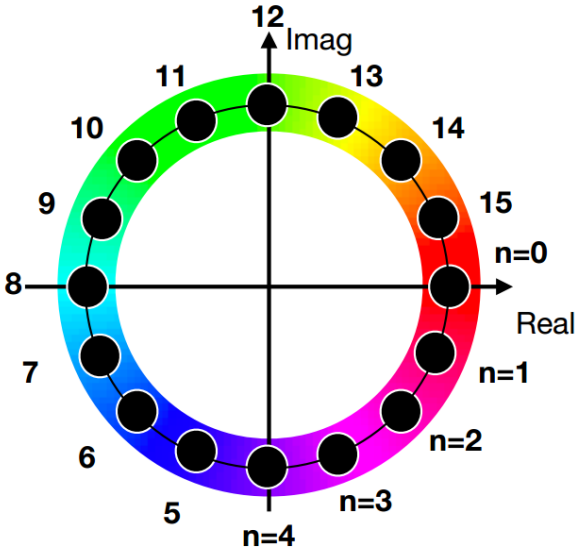
$$\exp\left(-2\pi j\frac{un}{N}\right)$$ For N=16

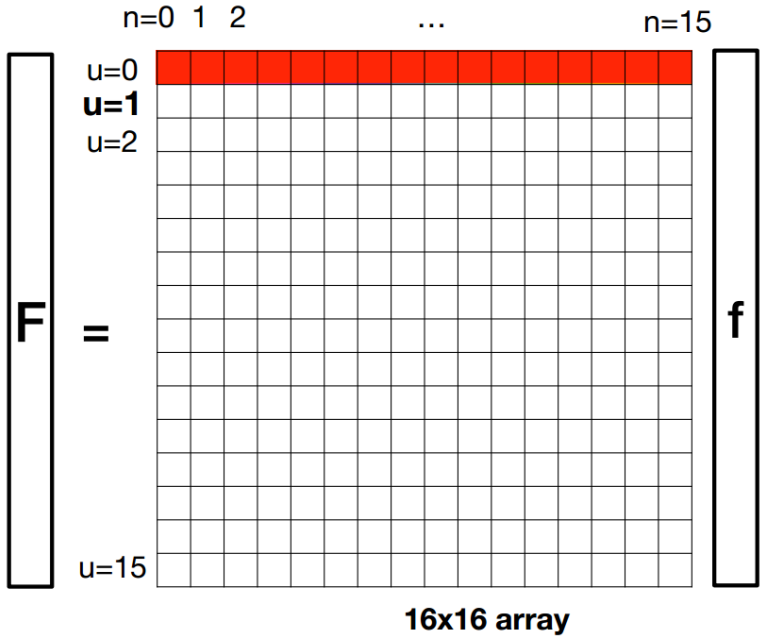




Visualizing the transform coefficients

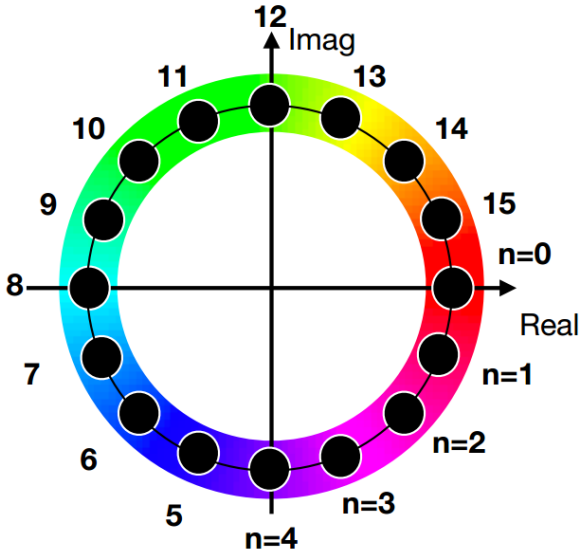
$$\exp\left(-2\pi j\frac{un}{N}\right)$$ For N=16

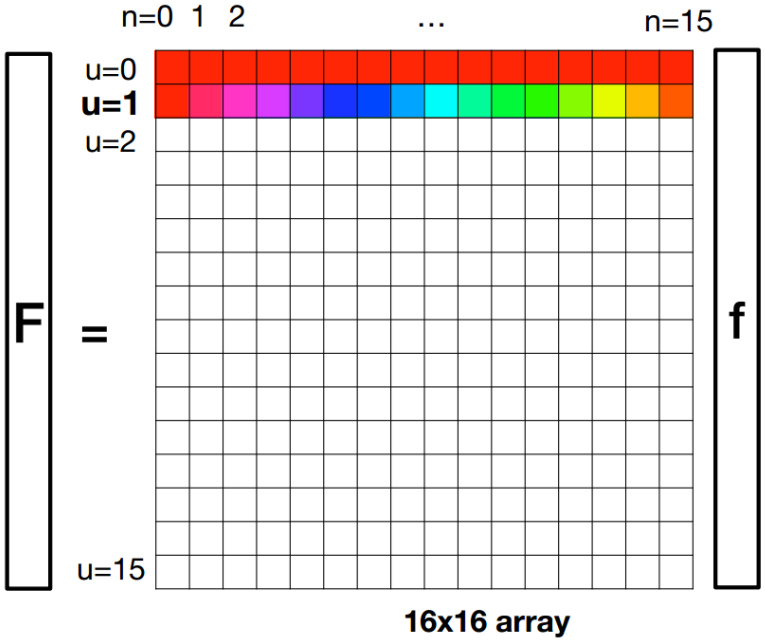




Visualizing the transform coefficients

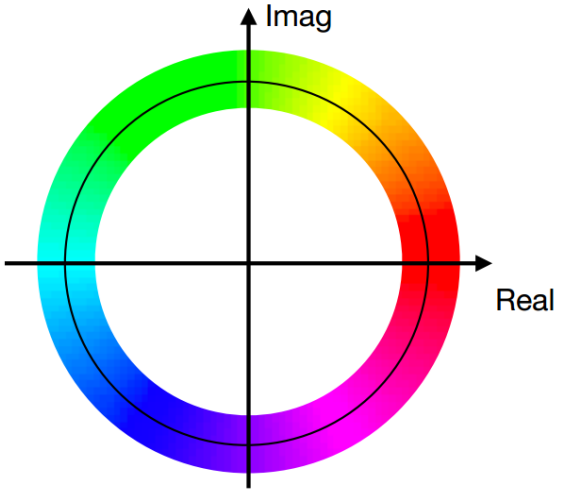
$$\exp\left(-2\pi j\frac{un}{N}\right)$$ For N=16

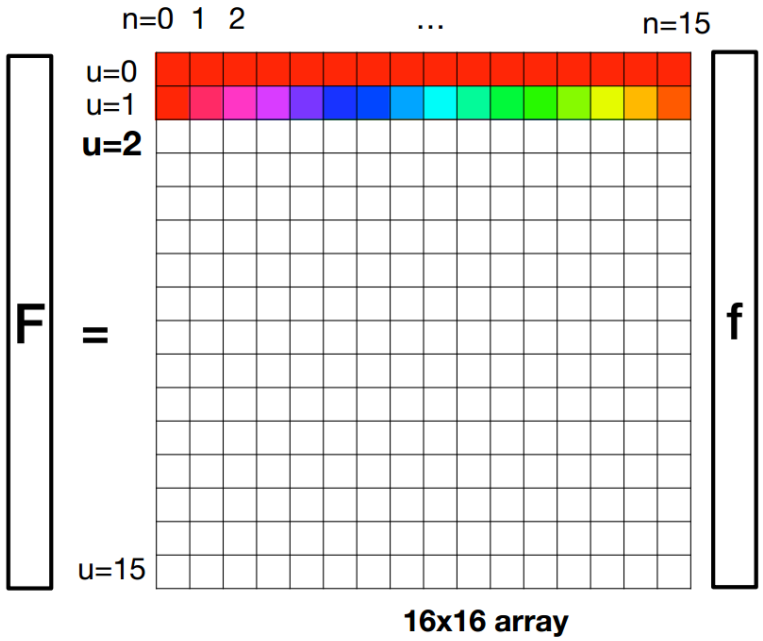




Visualizing the transform coefficients

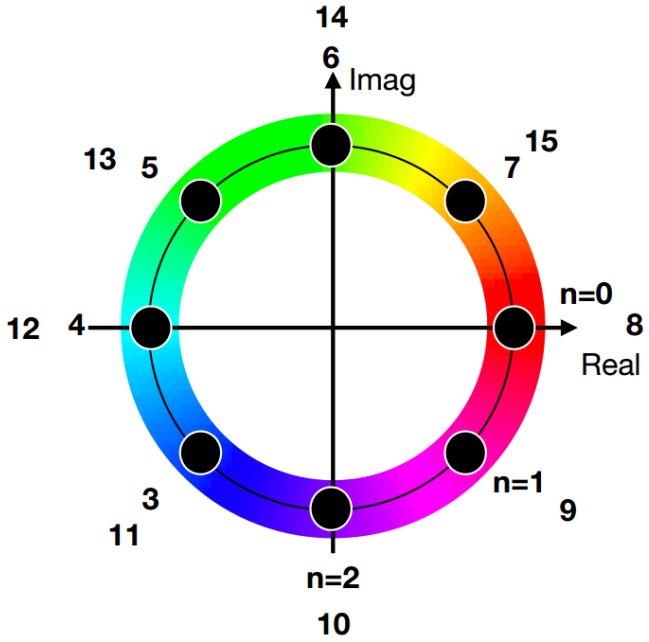
$$\exp\left(-2\pi j\frac{un}{N}\right)$$ For N=16

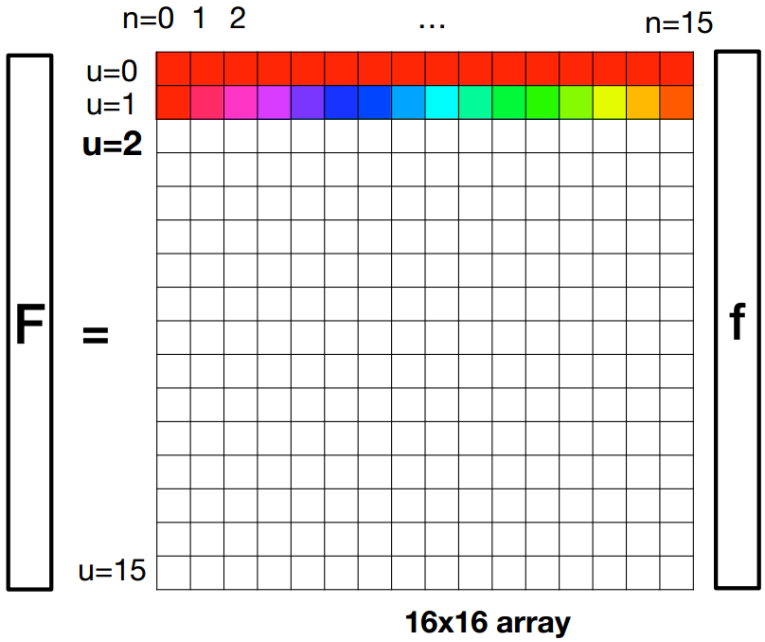




Visualizing the transform coefficients

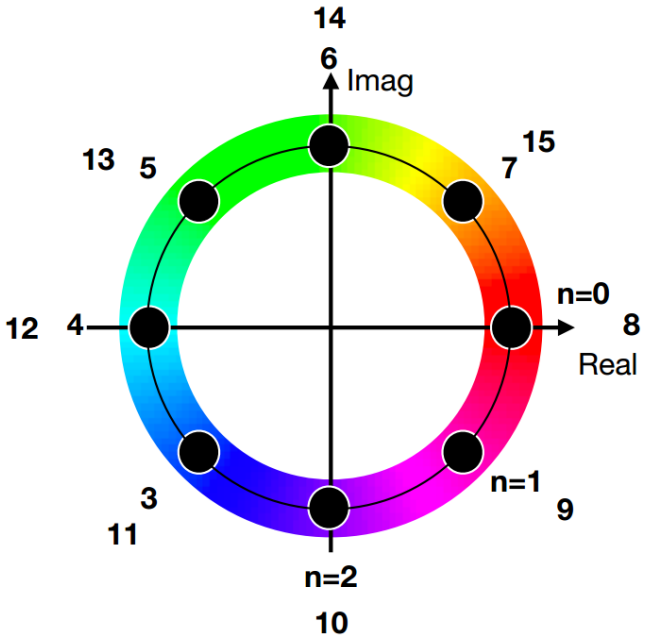
$$\exp\left(-2\pi j\frac{un}{N}\right)$$ For N=16

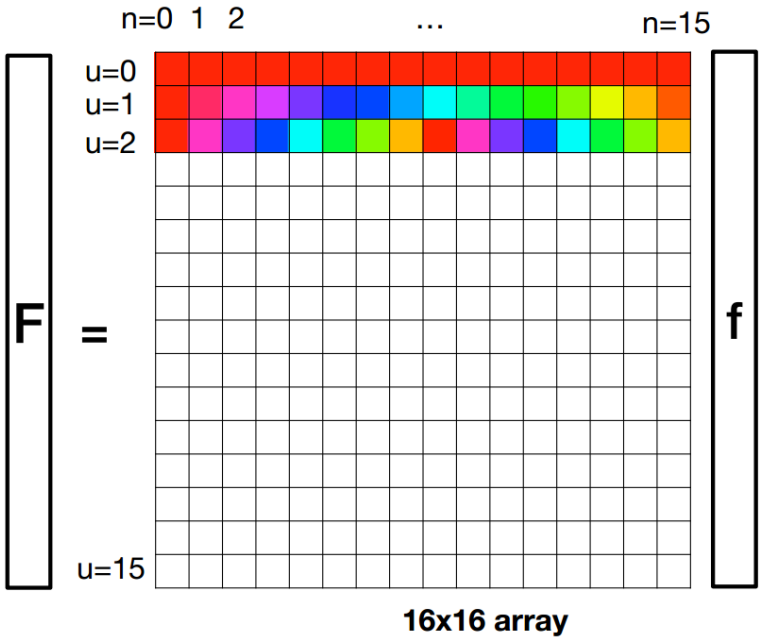




Visualizing the transform coefficients

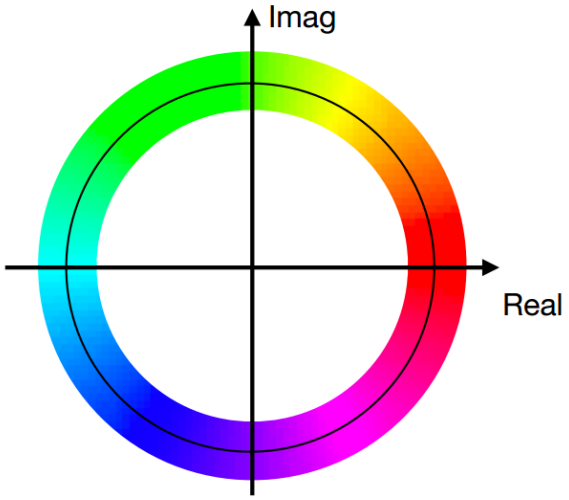
$$\exp\left(-2\pi j\frac{un}{N}\right)$$ For N=16

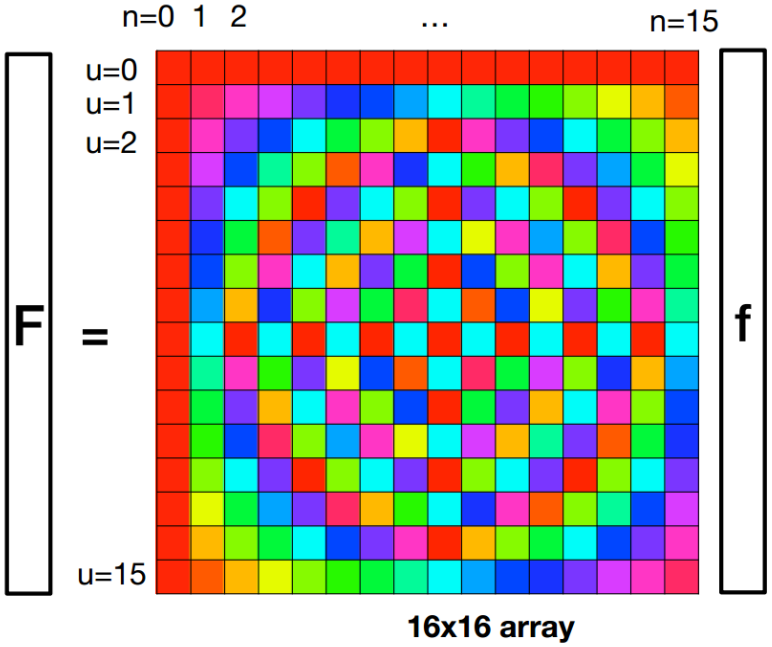




Visualizing the transform coefficients

$$\exp\left(-2\pi j\frac{un}{N}\right)$$ For N=16





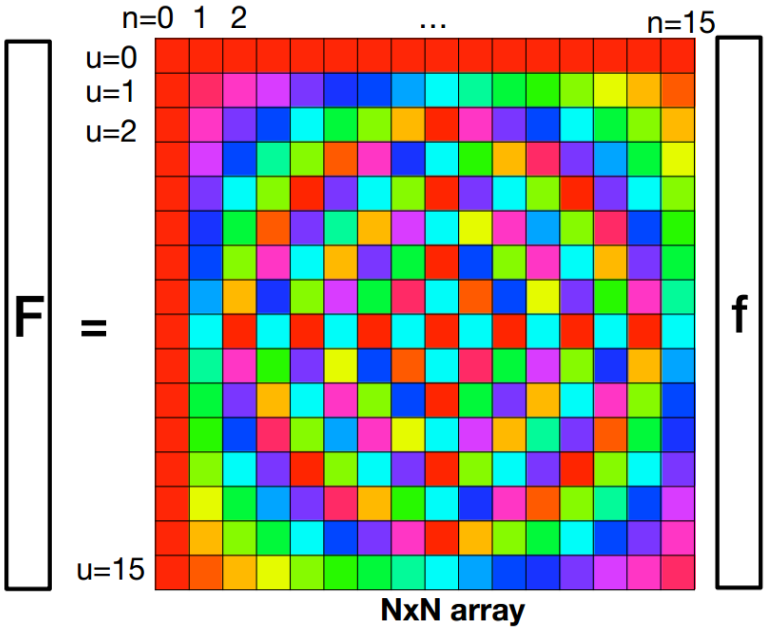
The inverse of the Discrete Fourier transform

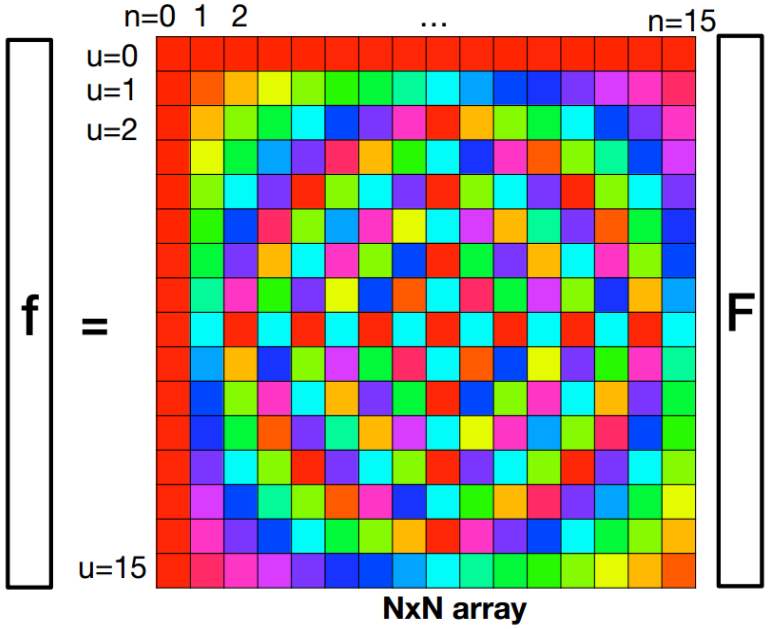
Discrete Fourier Transform (DFT):

$$F[u] = \sum_{n=0}^{N-1} f[n] \exp\left(-2\pi j \frac{un}{N}\right)$$

Its inverse:

$$f[n] = \frac{1}{N} \sum_{u=0}^{N-1} F[u] \exp\left(2\pi j \frac{un}{N}\right)$$



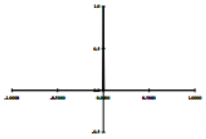
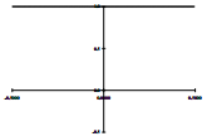
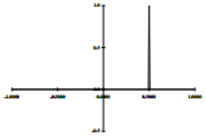
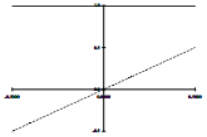
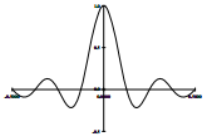
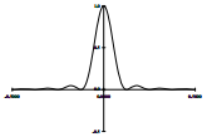
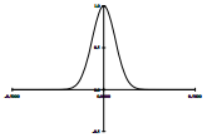
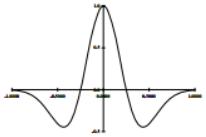
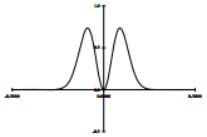
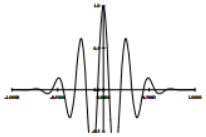
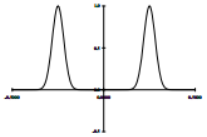
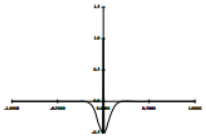
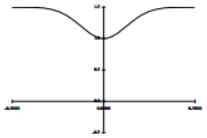
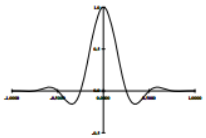
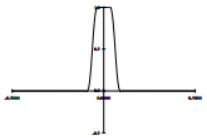


Fourier Transform

- Fourier transform stores the magnitude and phase at each frequency
  - Magnitude encodes how much signal there is at a particular frequency
  - Phase encodes spatial information (indirectly)
  - For mathematical convenience, this is often notated in terms of real and complex numbers

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

Euler's formula: $e^{inx} = \cos(nx) + i\sin(nx)$

Name	Signal		Transform	
impulse		$\delta(x)$	$\Leftrightarrow$	1 
shifted impulse		$\delta(x-u)$	$\Leftrightarrow$	$e^{-j\omega u}$ 
box filter		$\text{box}(x/a)$	$\Leftrightarrow$	$a\text{sinc}(a\omega)$ 
tent		$\text{tent}(x/a)$	$\Leftrightarrow$	$a\text{sinc}^2(a\omega)$ 
Gaussian		$G(x;\sigma)$	$\Leftrightarrow$	$\frac{\sqrt{2\pi}}{\sigma}G(\omega;\sigma^{-1})$ 
Laplacian of Gaussian		$(\frac{x^2}{\sigma^4}-\frac{1}{\sigma^2})G(x;\sigma)$	$\Leftrightarrow$	$-\frac{\sqrt{2\pi}}{\sigma}\omega^2 G(\omega;\sigma^{-1})$ 
Gabor		$\cos(\omega_0 x)G(x;\sigma)$	$\Leftrightarrow$	$\frac{\sqrt{2\pi}}{\sigma}G(\omega\pm\omega_0;\sigma^{-1})$ 
unsharp mask		$(1+\gamma)\delta(x)$ $-\gamma G(x;\sigma)$	$\Leftrightarrow$	$(1+\gamma)-$ $\frac{\sqrt{2\pi}\gamma}{\sigma}G(\omega;\sigma^{-1})$ 
windowed sinc		$\text{rcos}(x/(aW))$ $\text{sinc}(x/a)$	$\Leftrightarrow$	(see Figure 3.29) 

Summary of Fourier Transform

$s(t)$ transforms (continuous-time)

	Continuous frequency	Discrete frequencies
Transform	$S(f) \triangleq \int_{-\infty}^{\infty} s(t) \cdot e^{-i2\pi ft}\, dt$	$\overbrace{\frac{1}{P} \cdot S\left(\frac{k}{P}\right)}^{S[k]} \triangleq \frac{1}{P}\int_{-\infty}^{\infty} s(t) \cdot e^{-i2\pi \frac{k}{P}t}\, dt \equiv \frac{1}{P}\int_{P} s_P(t) \cdot e^{-i2\pi \frac{k}{P}t}\, dt$
Inverse	$s(t) = \int_{-\infty}^{\infty} S(f) \cdot e^{i2\pi ft}\, df$	$\underbrace{s_P(t) = \sum_{k=-\infty}^{\infty} S[k] \cdot e^{i2\pi \frac{k}{P}t}}_{\text{Poisson summation formula (Fourier series)}}$

$s(nT)$ transforms (discrete-time)

	Continuous frequency	Discrete frequencies
Transform	$\underbrace{\frac{1}{T}S_{\frac{1}{T}}(f) \triangleq \sum_{n=-\infty}^{\infty} s(nT) \cdot e^{-i2\pi fnT}}_{\text{Poisson summation formula (DTFT)}}$	$\overbrace{\frac{1}{T}S_{\frac{1}{T}}\left(\frac{k}{NT}\right)}^{S[k]} \triangleq \sum_{n=-\infty}^{\infty} s(nT) \cdot e^{-i2\pi \frac{kn}{N}} \equiv \underbrace{\sum_{n} s_P(nT) \cdot e^{-i2\pi \frac{kn}{N}}}_{\text{DFT}}$
Inverse	$s(nT) = T\int_{\frac{1}{T}} \frac{1}{T}S_{\frac{1}{T}}(f) \cdot e^{i2\pi fnT}\, df$ $\sum_{n=-\infty}^{\infty} s(nT) \cdot \delta(t-nT) = \underbrace{\int_{-\infty}^{\infty} \frac{1}{T}\, S_{\frac{1}{T}}(f) \cdot e^{i2\pi ft}\, df}_{\text{inverse Fourier transform}}$	$s_P(nT) = \overbrace{\frac{1}{N}\sum_{k} S[k] \cdot e^{i2\pi \frac{kn}{N}}}^{\text{inverse DFT}} = \frac{1}{P}\sum_{k} S_{\frac{1}{T}}\left(\frac{k}{P}\right) \cdot e^{i2\pi \frac{kn}{N}}$

Today's Class

- Frequency domain and Fourier transform
- 1D Discrete Fourier Transform
- **2D Image Fourier Transform**
- Sampling
- Hybrid Image

For images, the 2D DFT

1D Discrete Fourier Transform (DFT) transforms a signal $f[n]$ into $F[u]$ as:

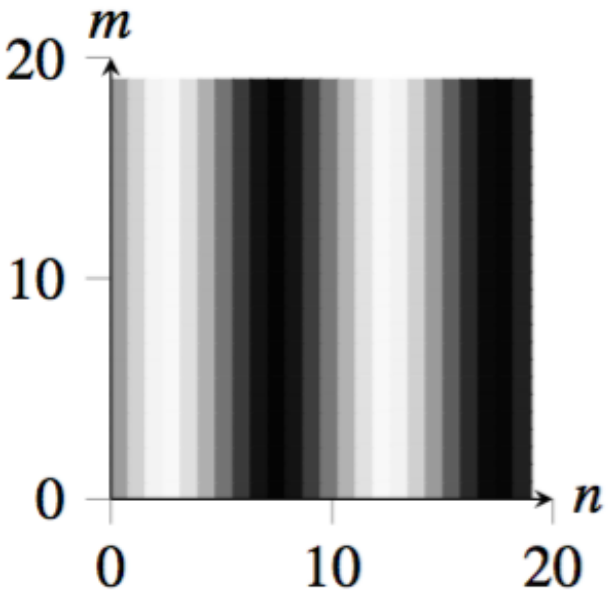
$$F[u] = \sum_{n=0}^{N-1} f[n] \exp\left(-2\pi j \frac{un}{N}\right)$$

2D Discrete Fourier Transform (DFT) transforms an image $f[n,m]$ into $F[u,v]$ as:

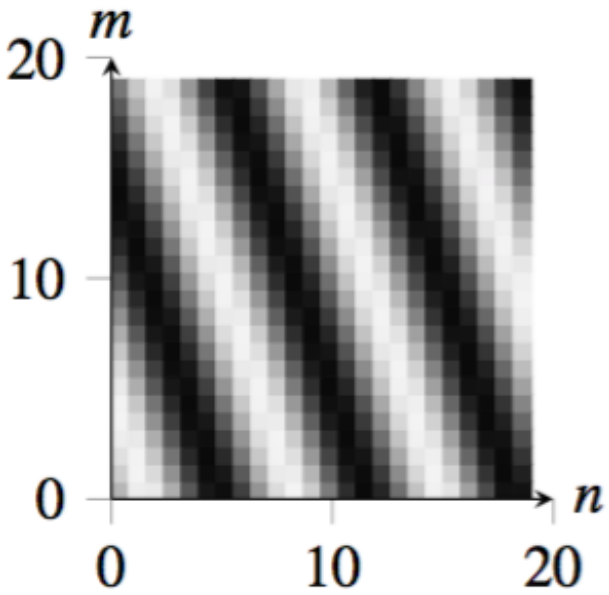
$$F[u,v] = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[n,m] \exp\left(-2\pi j \left(\frac{un}{N} + \frac{vm}{M}\right)\right)$$

Waves in 2D

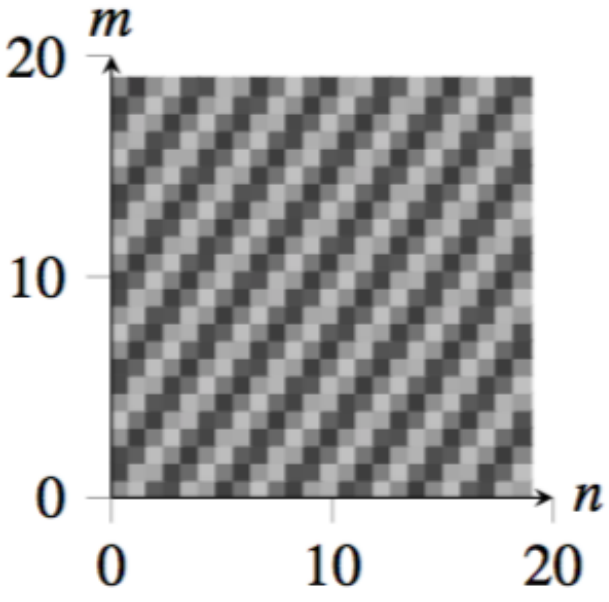
$$s_{u,v}[n,m] = A\sin\left(2\pi\left(\frac{un}{N}+\frac{vm}{M}\right)\right) \qquad c_{u,v}[n,m] = A\cos\left(2\pi\left(\frac{un}{N}+\frac{vm}{M}\right)\right)$$



u = 2, v = 0



u = 3, v = 1

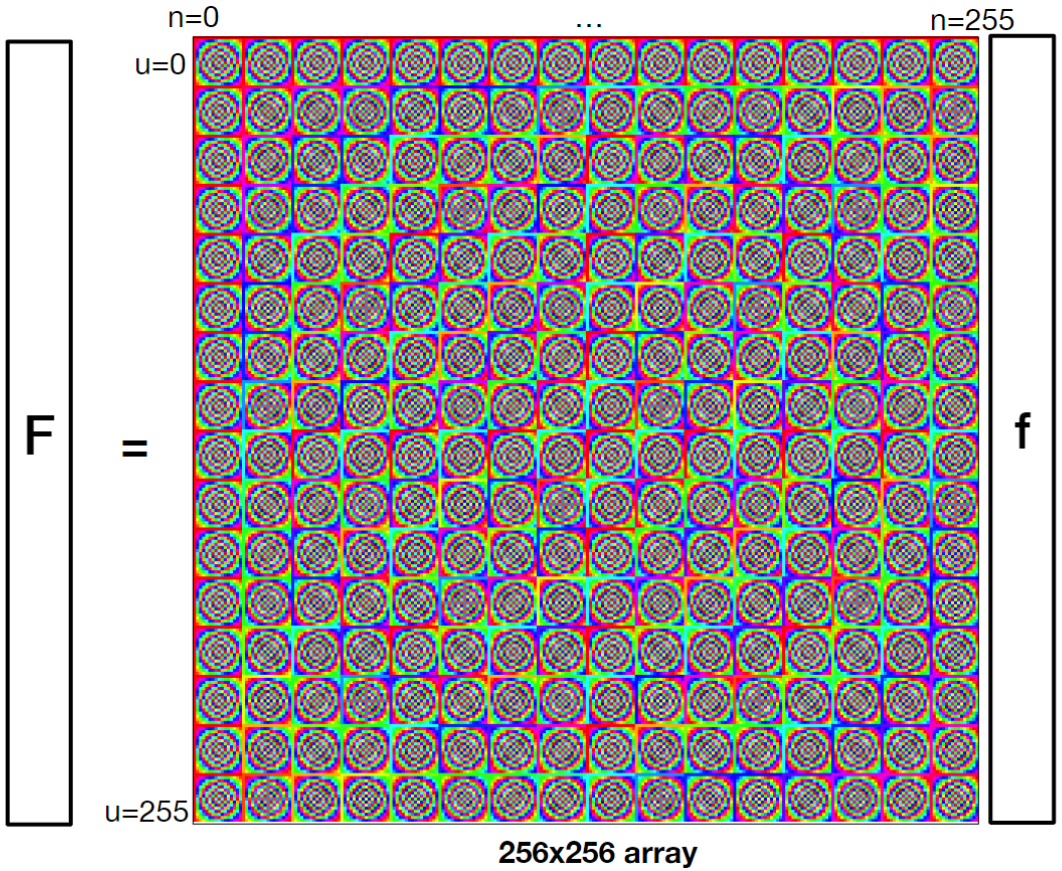


u = 7, v = −5

Visualizing the 2D DFT coefficients

$$F[u,v] = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} f[n,m] \exp\left(-2\pi j\left(\frac{un}{N}+\frac{vm}{M}\right)\right)$$

For N=M=16



Visualizing the image Fourier transform

$$F[u,v] = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[n,m] \exp\left(-2\pi j \left(\frac{un}{N} + \frac{vm}{M}\right)\right)$$

The values of $F[u,v]$ are complex.

Using the real and imaginary components:

$$F[u,v] = Re\{F[u,v]\} + j\,Imag\{F[u,v]\}$$

Or using a polar decomposition:

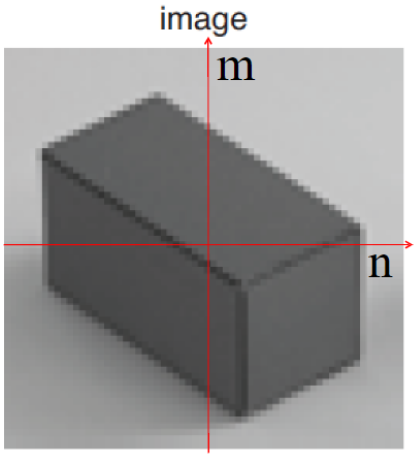
$$F[u,v] = A[u,v] \exp(j\theta[u,v])$$

Amplitude

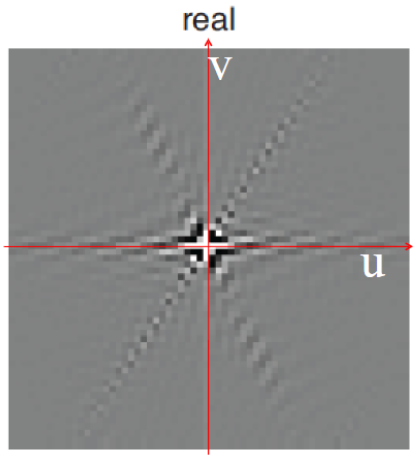
Phase

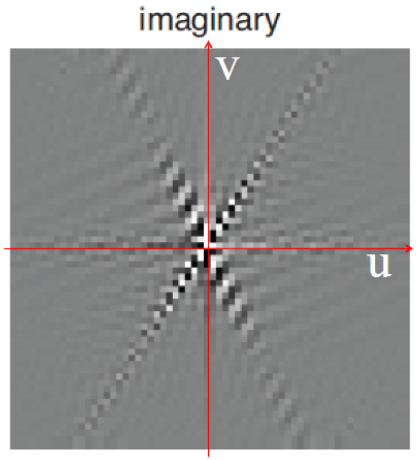
Visualizing the image Fourier transform

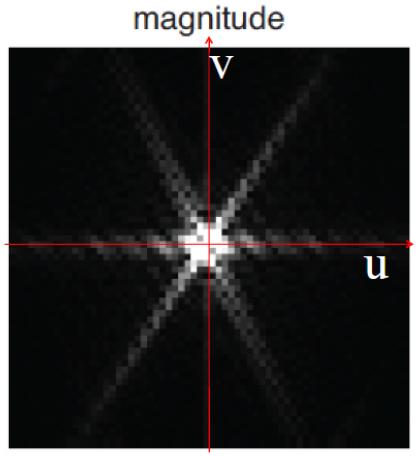
$f[n, m]$

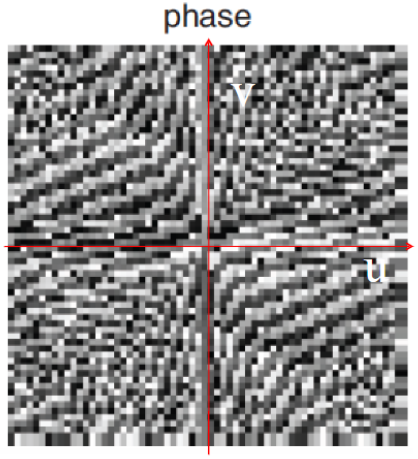


$F[u, v]$

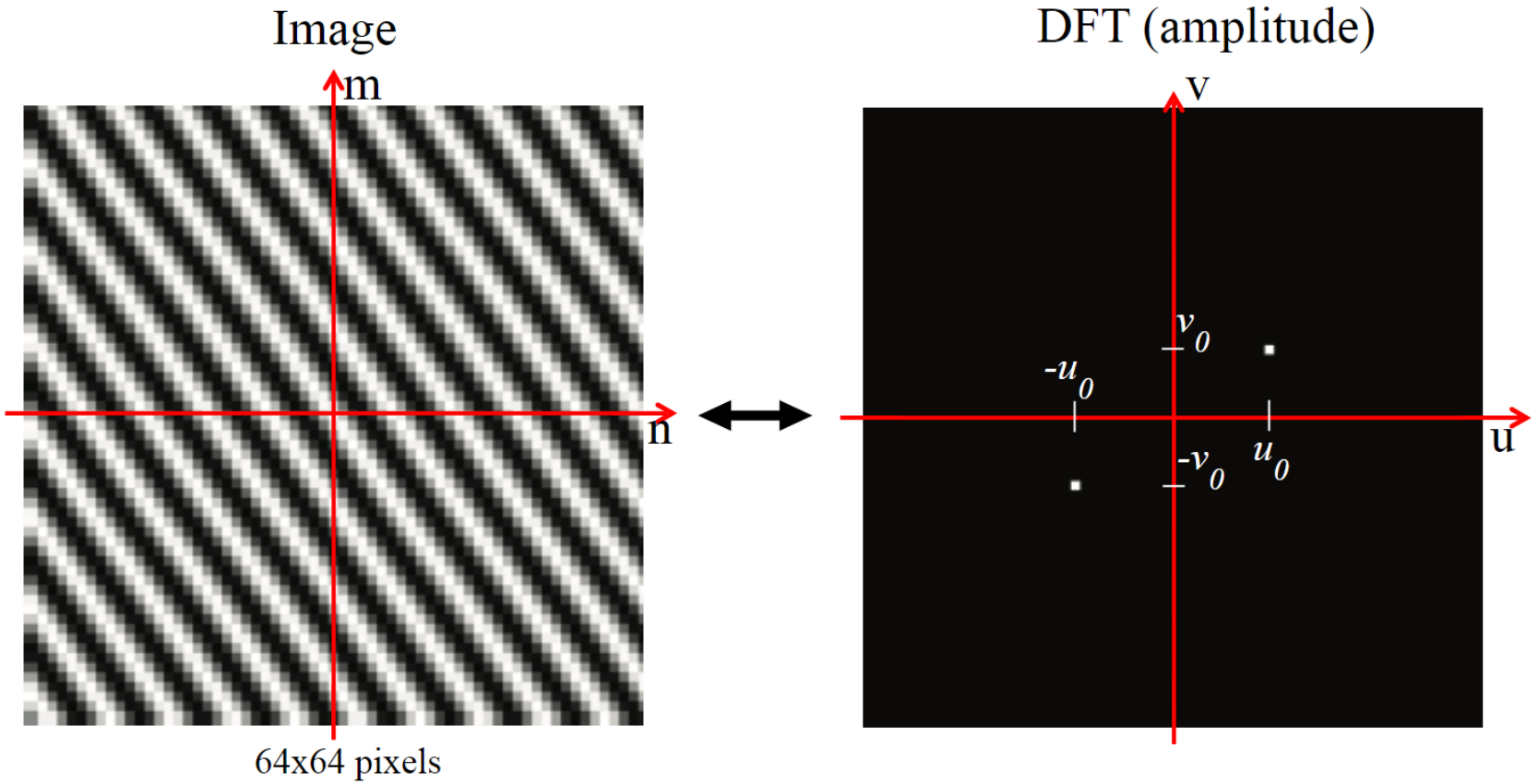






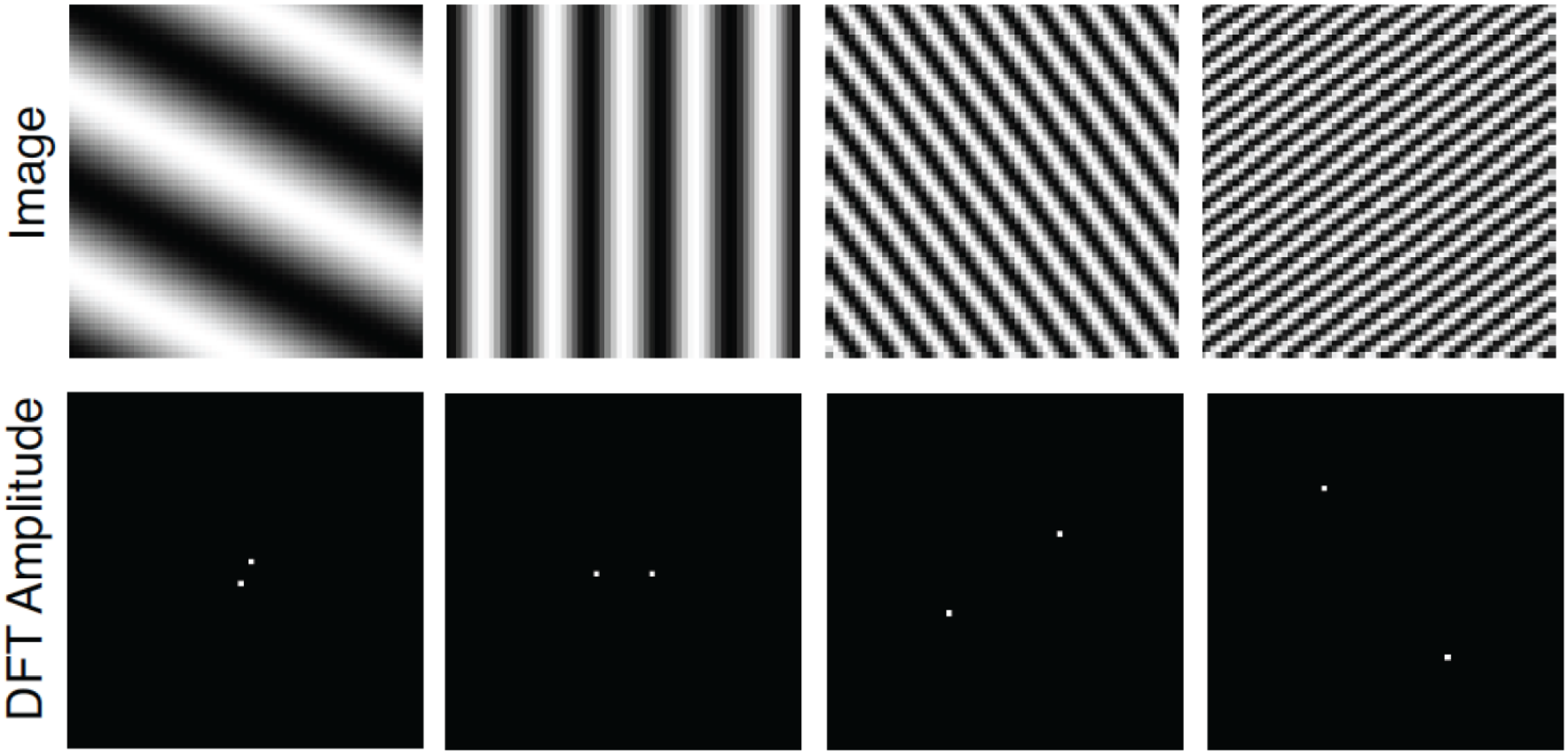


Simple Fourier transforms



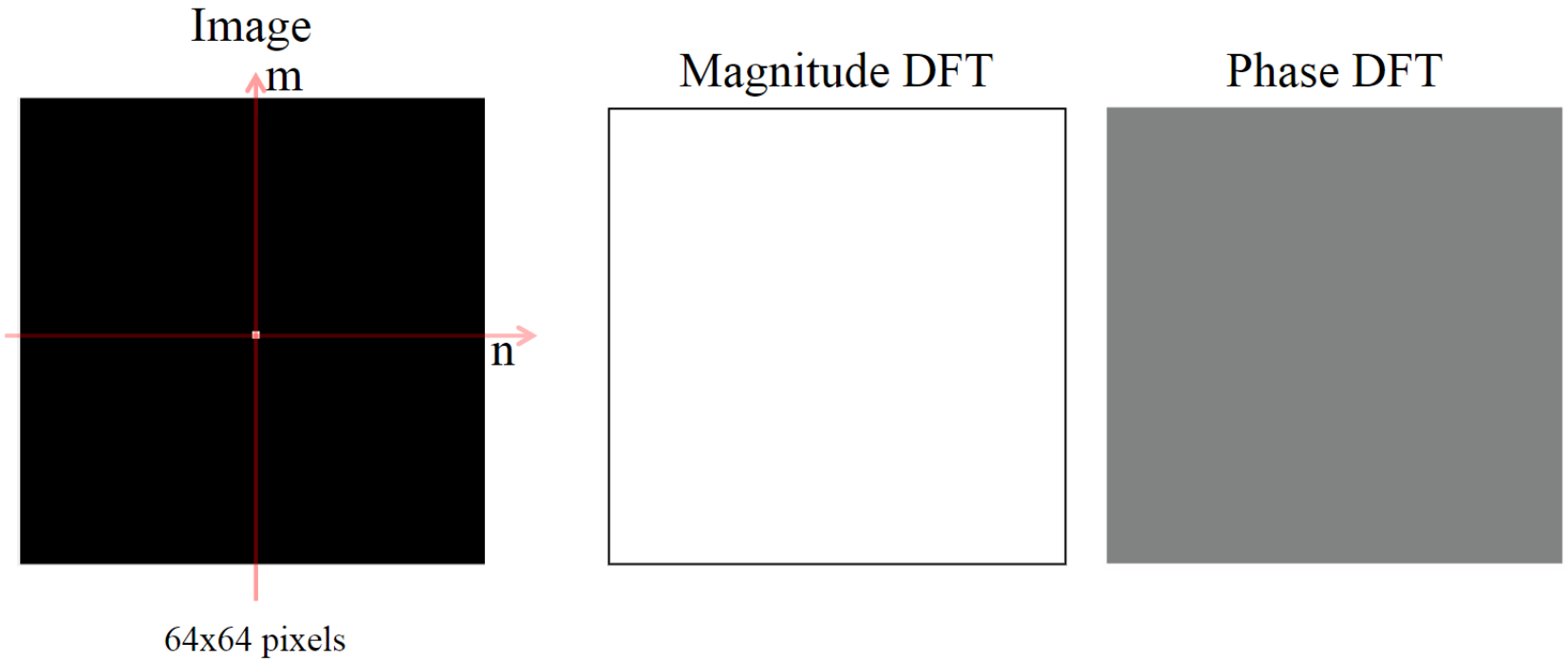
$$\cos\left(2\pi\left(\frac{u_0\,n}{N}+\frac{v_0\,m}{M}\right)\right) \longleftrightarrow \frac{1}{2}\left(\delta\left[u-u_0,v-v_0\right]+\delta\left[u+u_0,v+v_0\right]\right)$$

Simple Fourier transforms

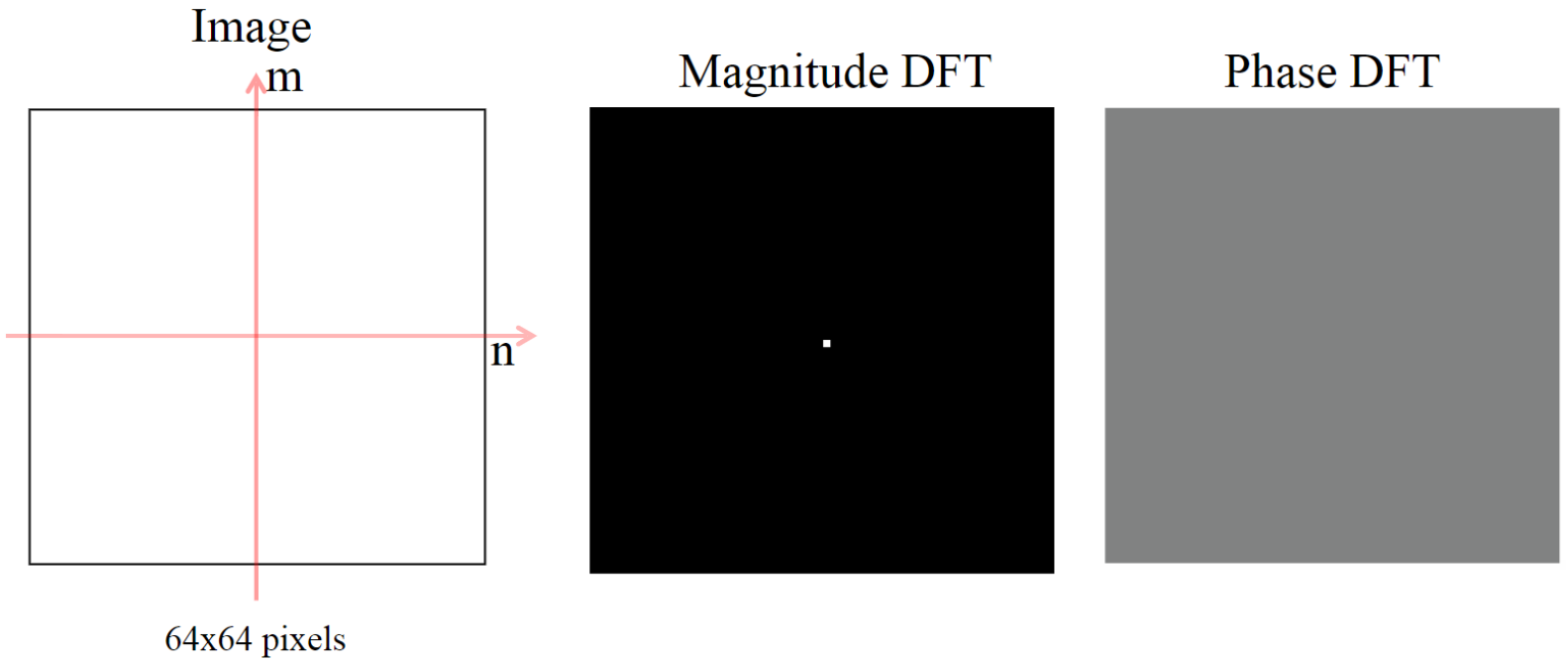


Images are 64x64 pixels. The wave is a cosine, therefore DFT phase is zero.

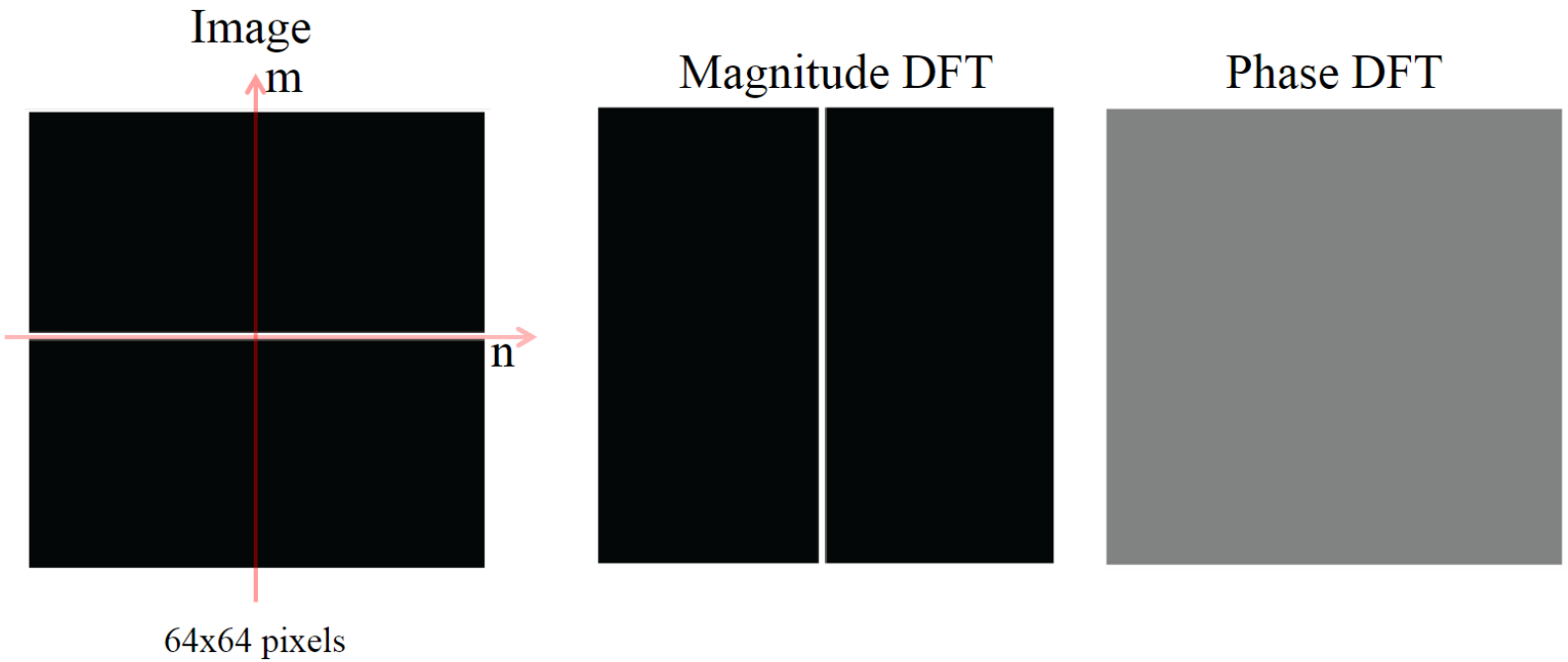
Some important Fourier transforms



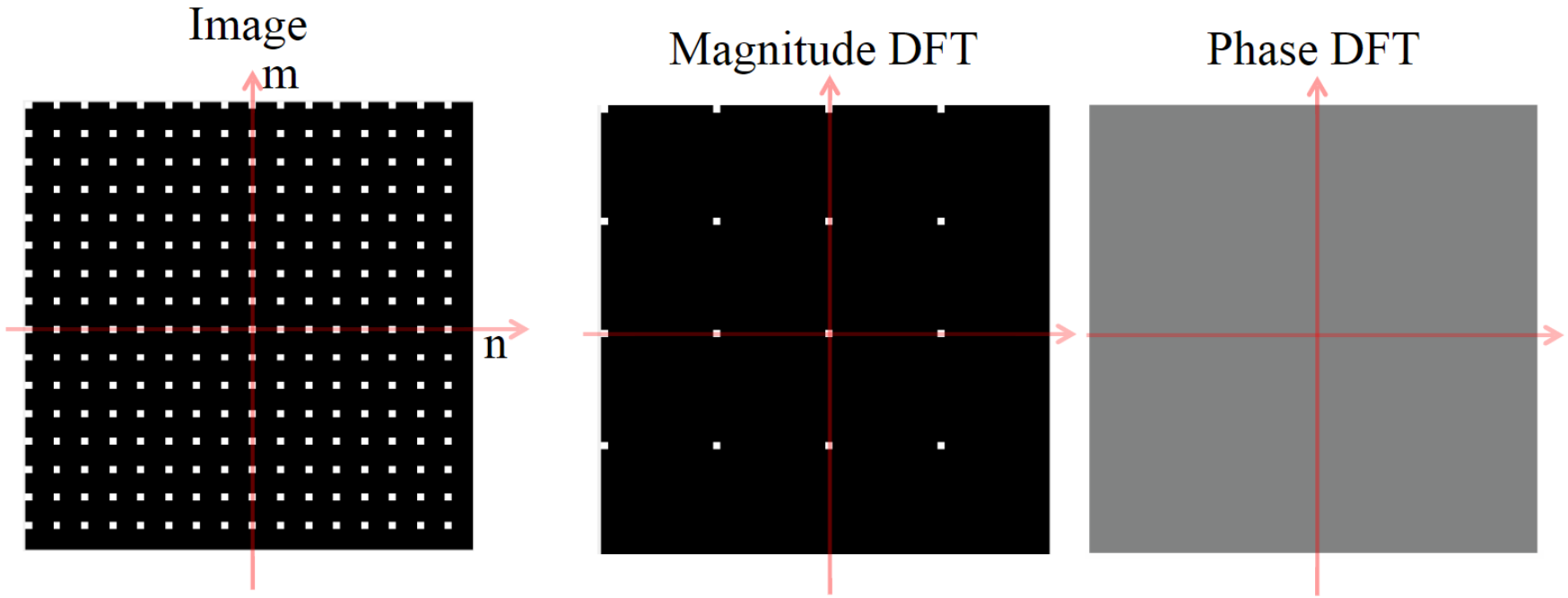
Some important Fourier transforms



Some important Fourier transforms



Some important Fourier transforms

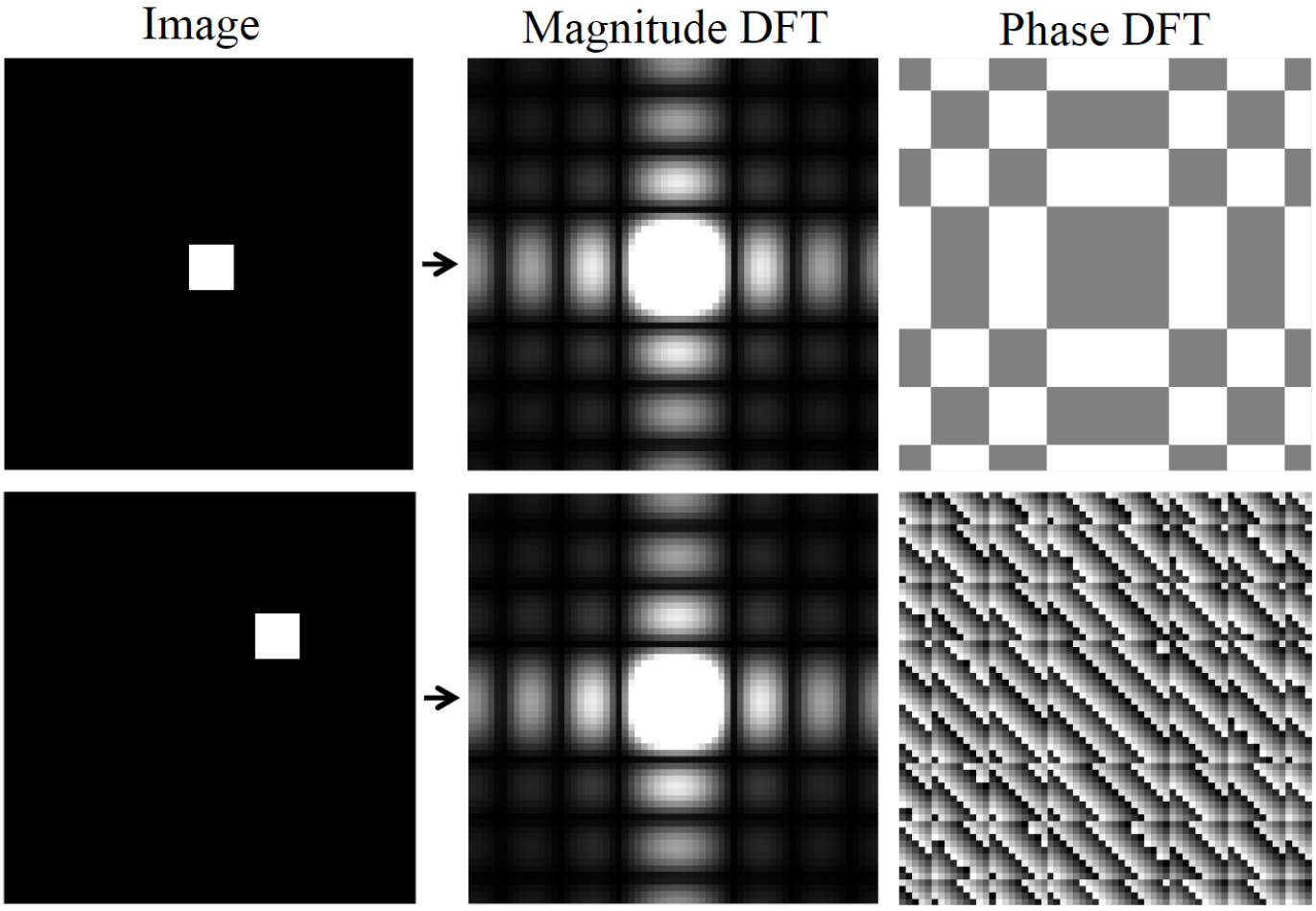


64x64 pixels

when M and N are divisible by k

$$\delta_k[n,m] = \sum_{s=0}^{N/k-1} \sum_{r=0}^{M/k-1} \delta[n - sk, m - rk] \quad \longleftrightarrow \quad \Delta_k[u,v] = \frac{NM}{k^2} \sum_{s=0}^{k-1} \sum_{r=0}^{k-1} \delta\left[u - s\frac{N}{k}, v - r\frac{M}{k}\right]$$

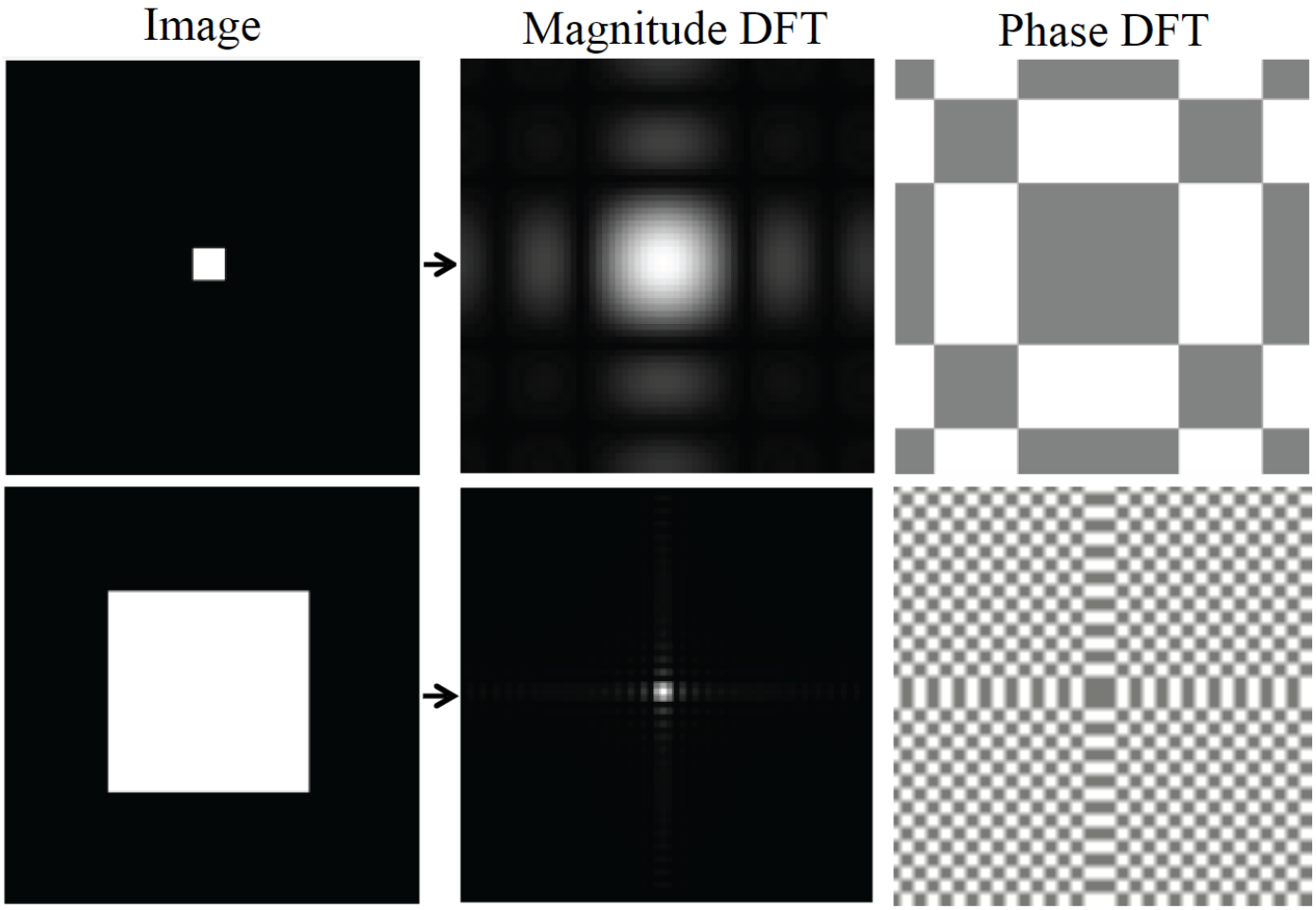
Some important Fourier transforms



Translation

Shifts of an image only produce changes on the phase of the DFT.

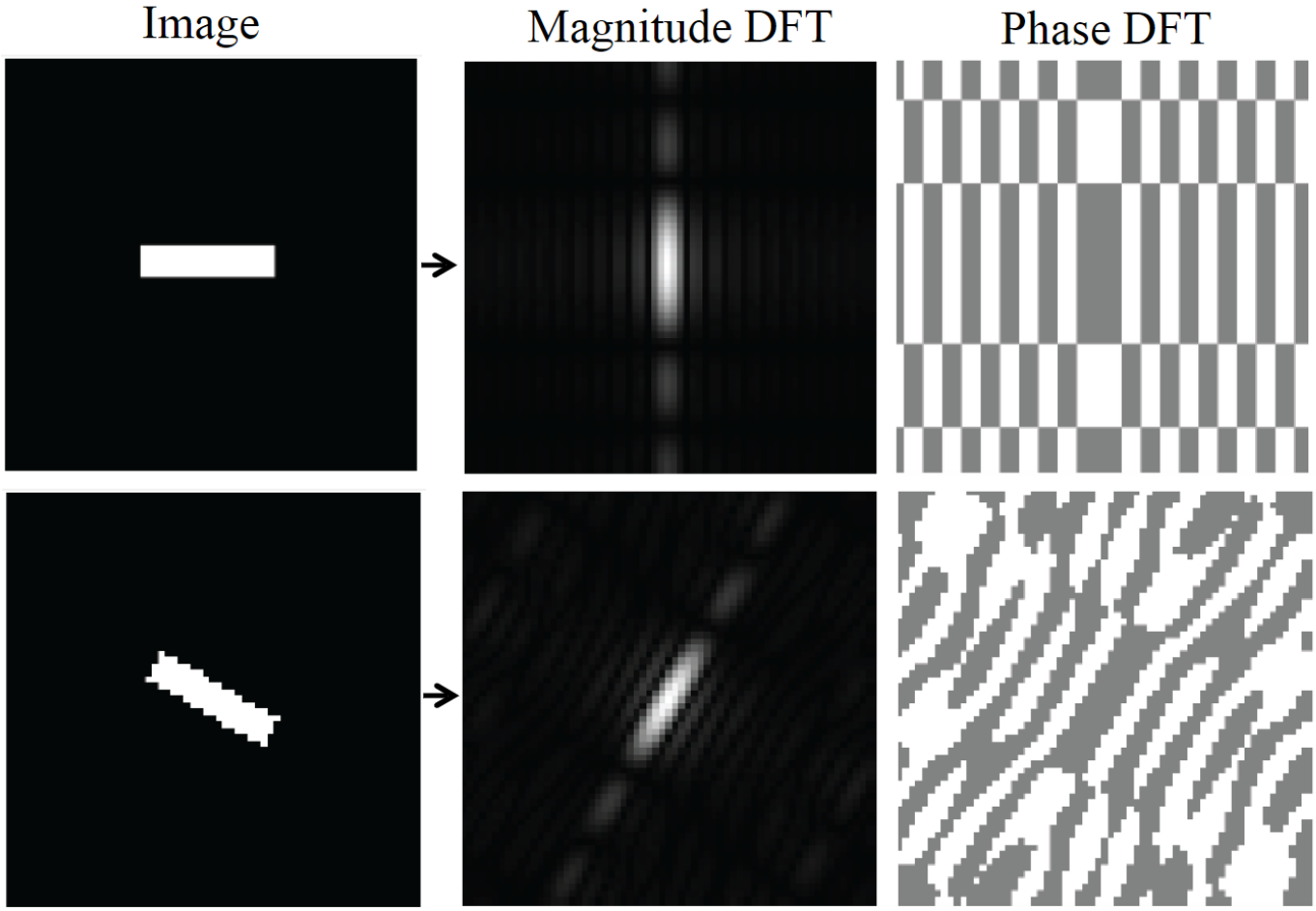
Some important Fourier transforms



Scale

Small image details produce content in high spatial frequencies

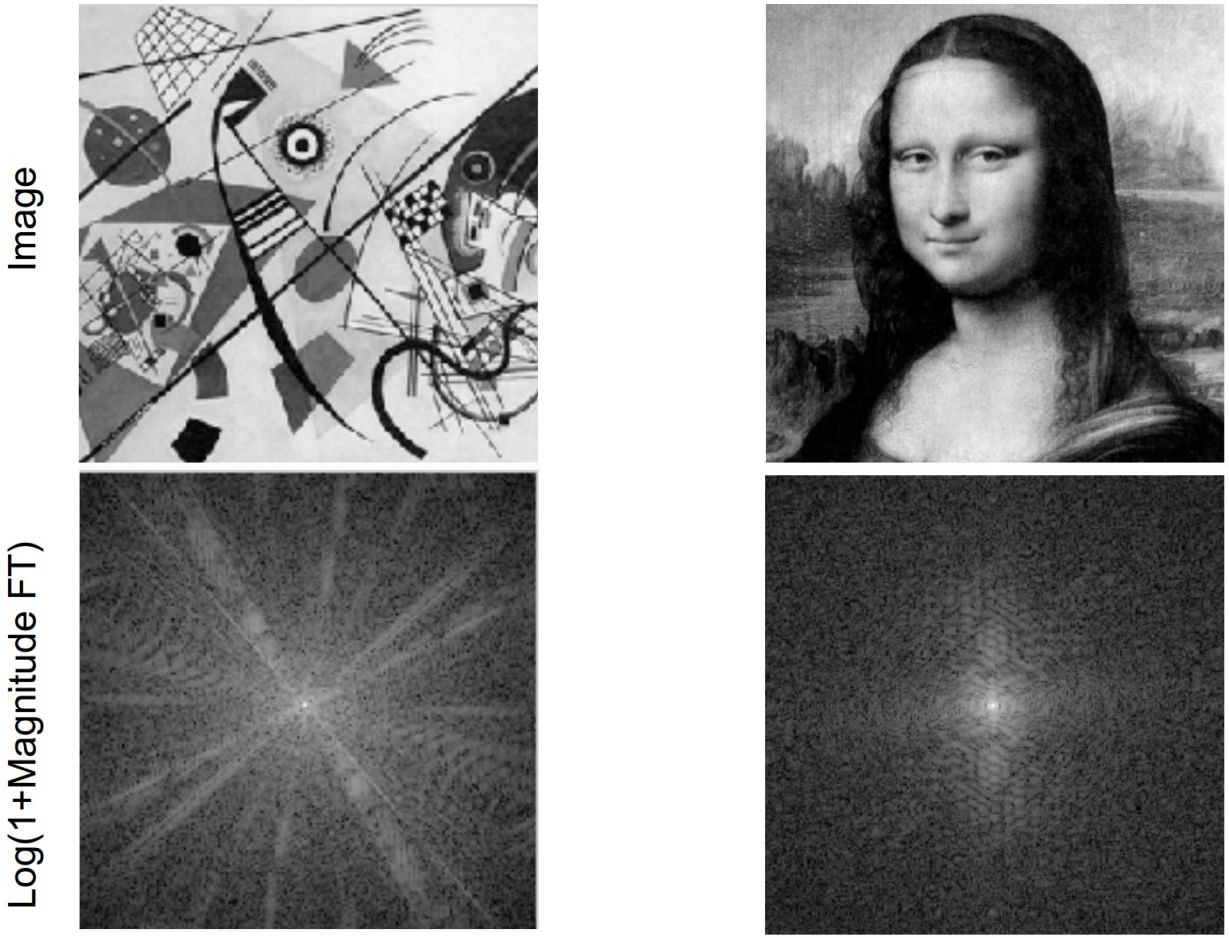
Some important Fourier transforms



Orientation

A line transforms to a line oriented perpendicularly to the first.

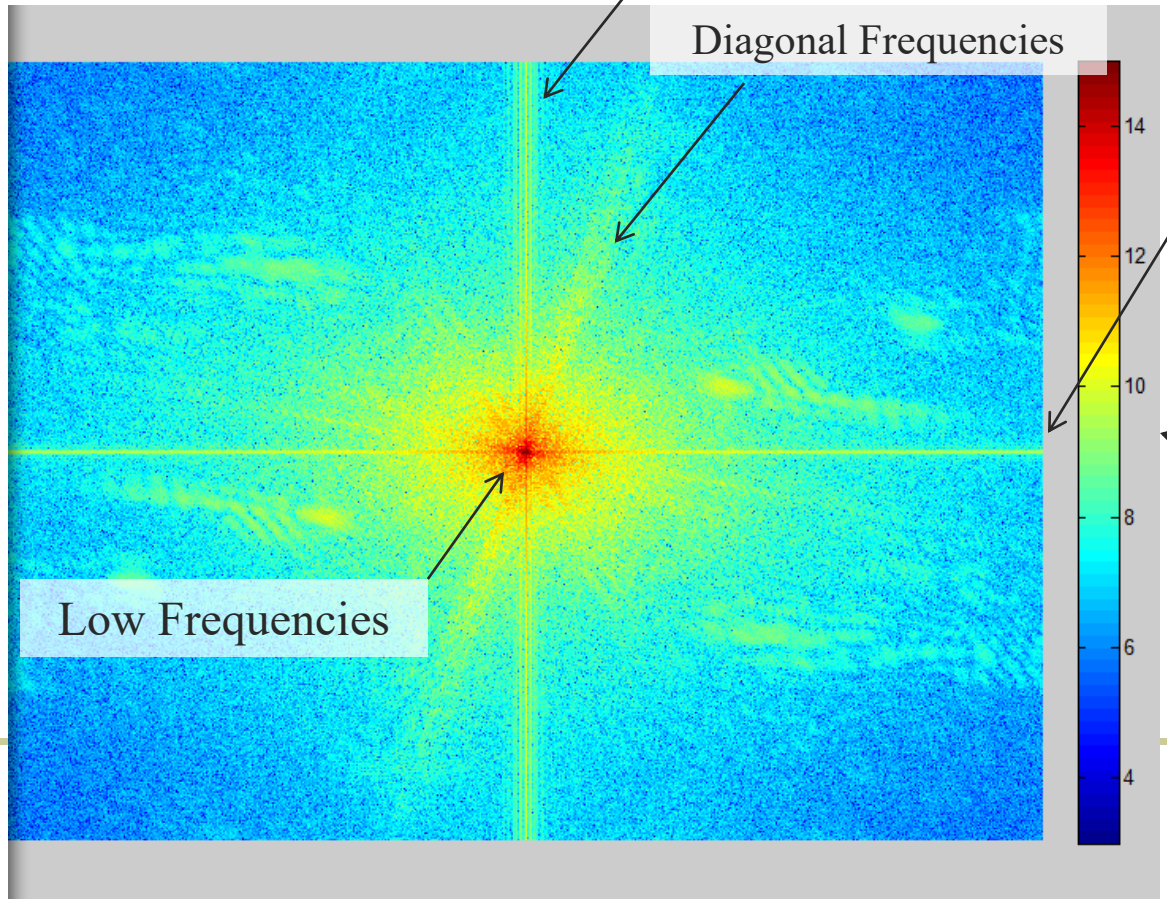
The Fourier Transform of some important images



Strong Vertical Frequency
(Sharp Horizontal Edge)

Diagonal Frequencies

Strong Horz. Frequency
(Sharp Vert. Edge)



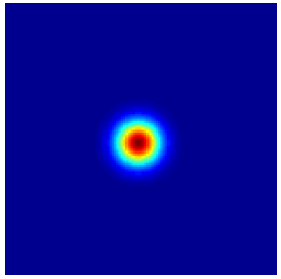
Log Magnitude

Low Frequencies

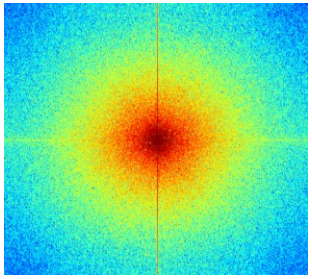
Question

Match the spatial domain image to the Fourier magnitude image

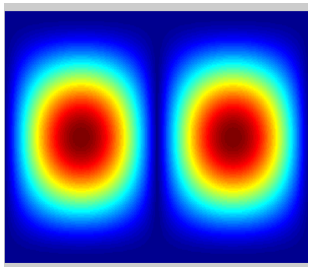
1



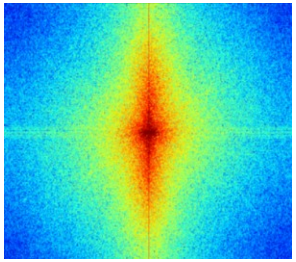
2



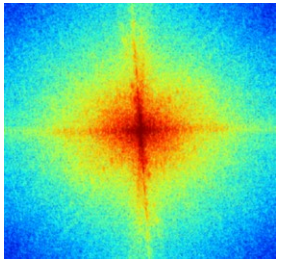
3



4



5

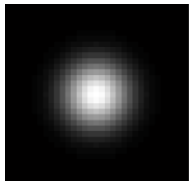


A

B

C

D



E

The inverse Discrete Fourier transform

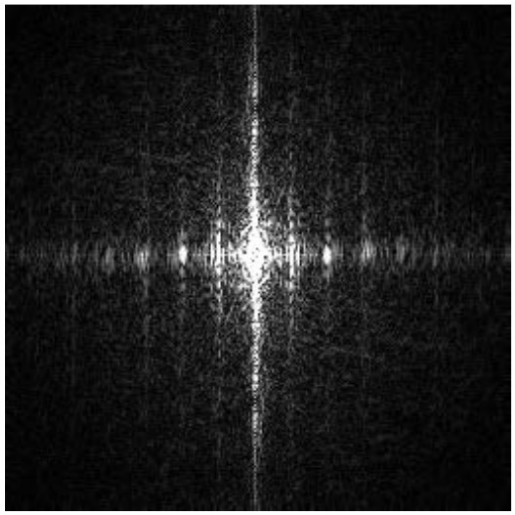
2D Discrete Fourier Transform (DFT) transforms an image *f* [*n*,*m*] into *F* [*u*,*v*] as:

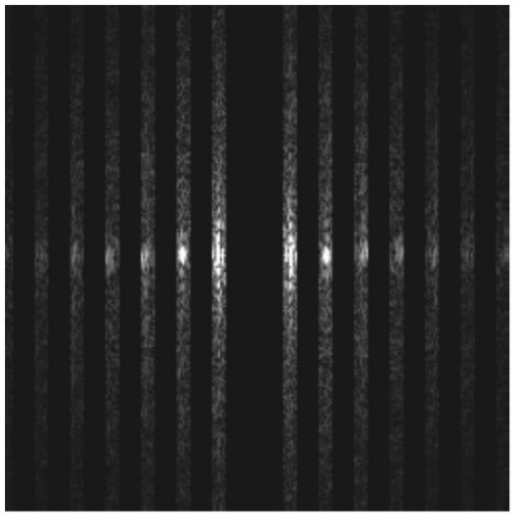
$$F[u,v] = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} f[n,m] \exp\left(-2\pi j\left(\frac{un}{N}+\frac{vm}{M}\right)\right)$$

The inverse of the 2D DFT is:

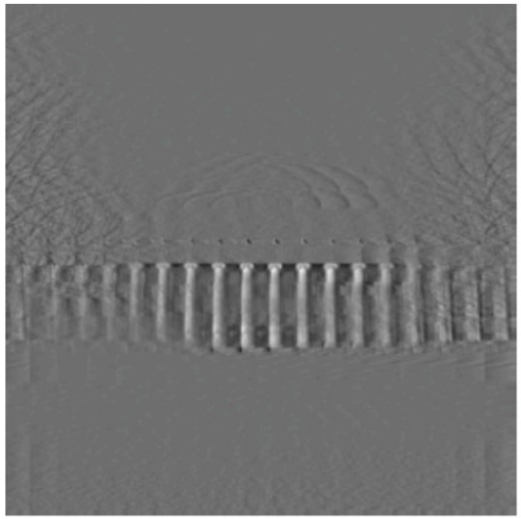
$$f[n,m] = \frac{1}{NM}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} F[u,v] \exp\left(+2\pi j\left(\frac{un}{N}+\frac{vm}{M}\right)\right)$$

DFT →

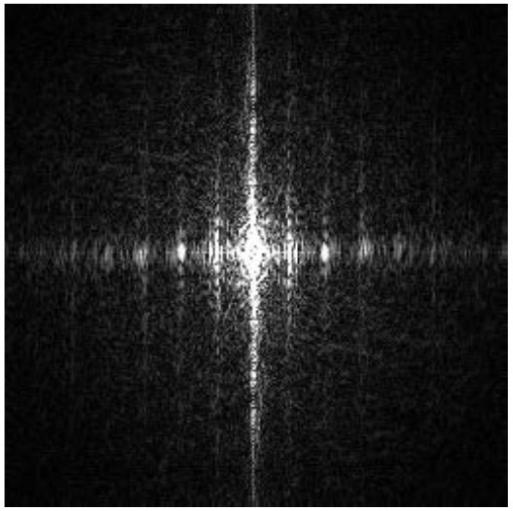


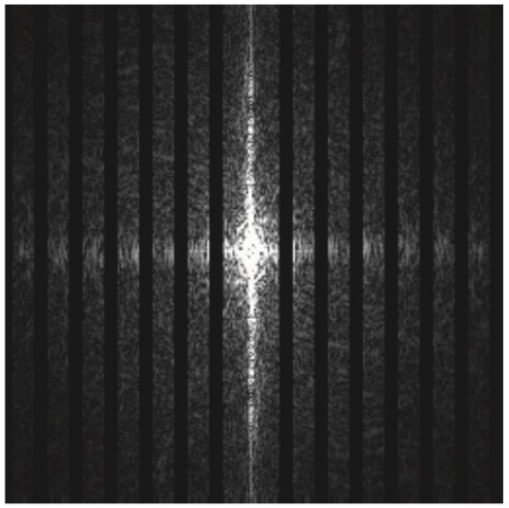


DFT^{-1} →



DFT





DFT^{-1}

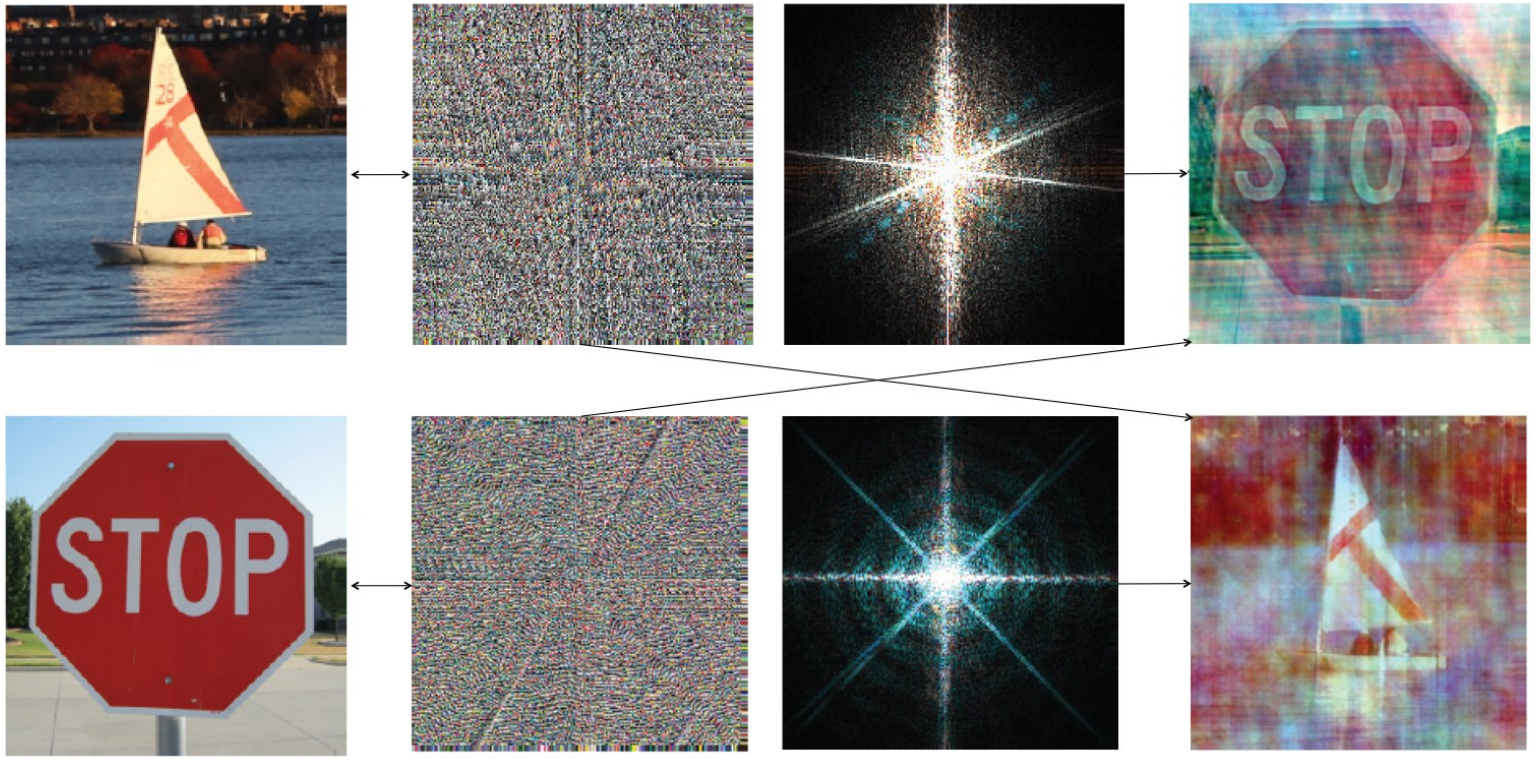
Questions

Which has more information, the phase or the magnitude?

What happens if you take the phase from one image and combine it with the magnitude from another image?

Phase and Magnitude

$$F[u,v] = A[u,v]\exp(j\theta[u,v])$$



Each color channel is processed in the same way.

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!

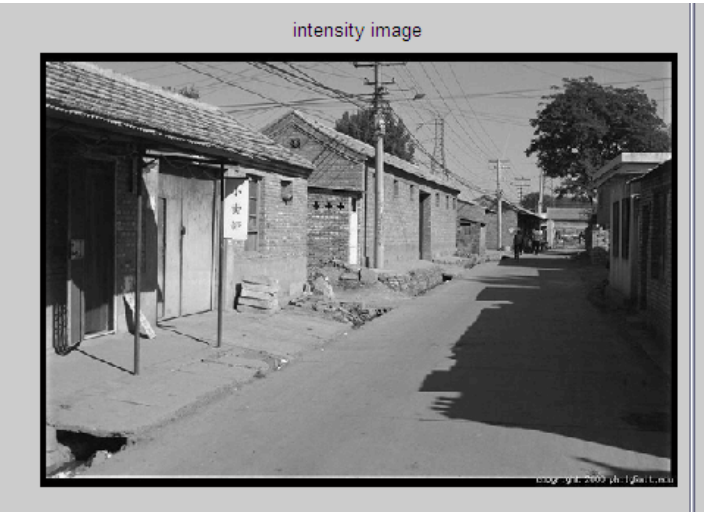
Properties of Fourier Transforms

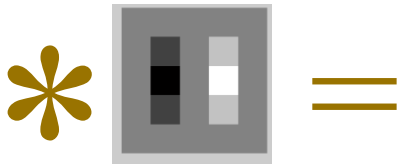
- Linearity $\mathcal{F}[ax(t) + by(t)] = a\mathcal{F}[x(t)] + b\mathcal{F}[y(t)]$
- Fourier transform of a real signal is symmetric about the origin
- The energy of the signal is the same as the energy of its Fourier transform

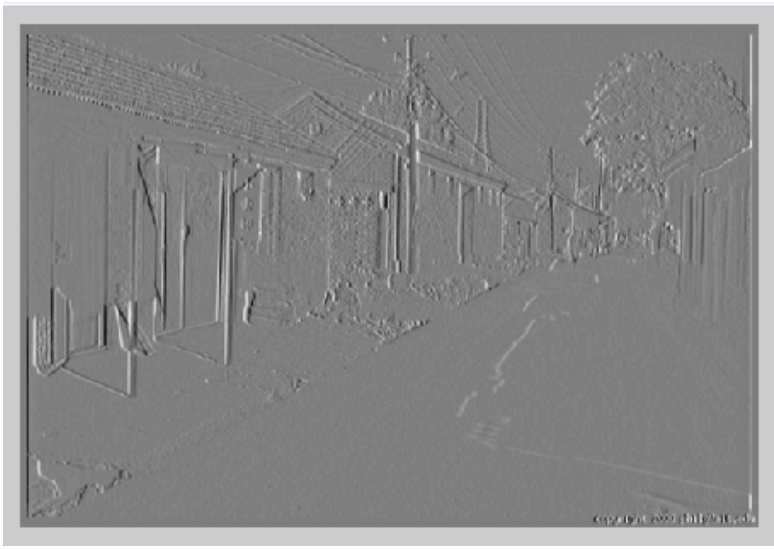
See Szeliski Book (3.4)

Filtering in spatial domain

1	0	-1
2	0	-2
1	0	-1

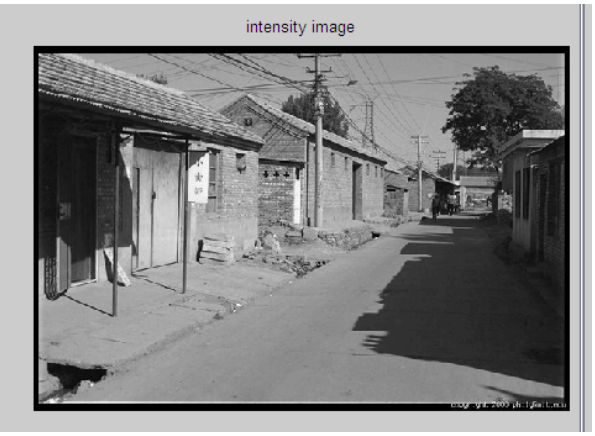




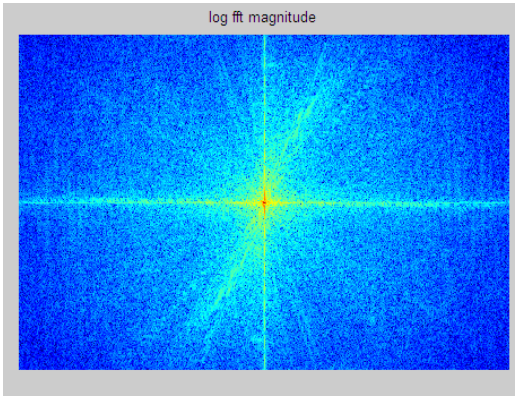


Filtering in frequency domain

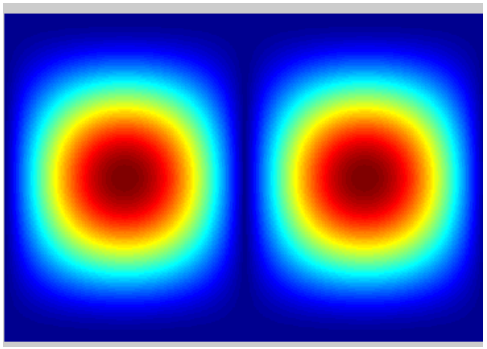
FFT



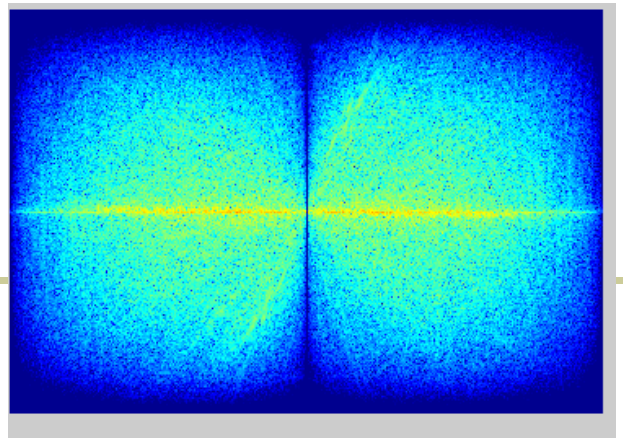
FFT



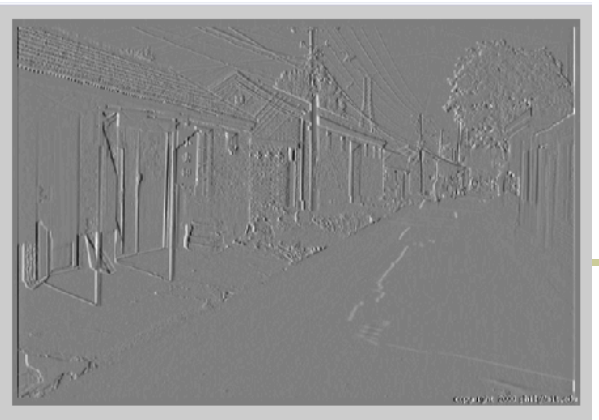
×



=



Inverse FFT



FFT in Matlab

- Filtering with fft

```
im = ... % "im" should be a gray-scale floating point image
[imh, imw] = size(im);
fftsize = 1024; % should be order of 2 (for speed) and include padding
im_fft = fft2(im, fftsize, fftsize); % 1) fft im with padding
hs = 50; % filter half-size
fil = fspecial('gaussian', hs*2+1, 10);
fil_fft = fft2(fil, fftsize, fftsize); % 2) fft fil, pad to same size as image
im_fil_fft = im_fft .* fil_fft; % 3) multiply fft images
im_fil = ifft2(im_fil_fft); % 4) inverse fft2
im_fil = im_fil(1+hs:size(im,1)+hs, 1+hs:size(im, 2)+hs); % 5) remove padding
```

- Displaying with fft

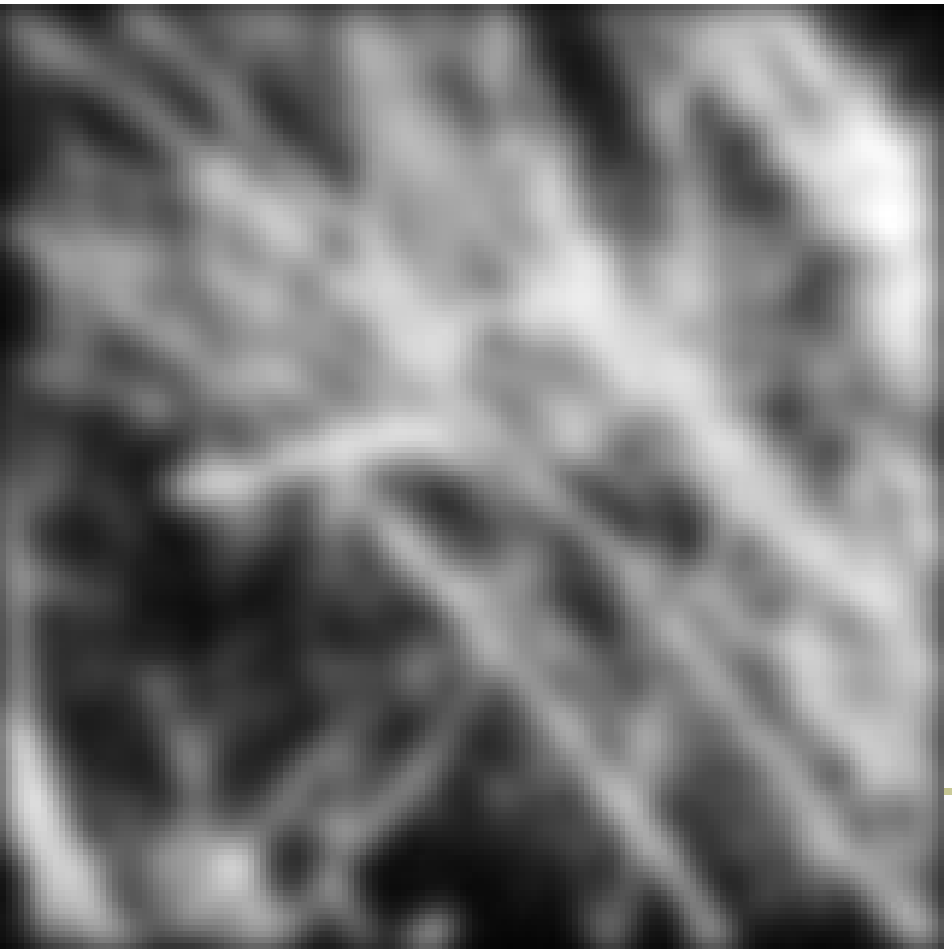
```
figure(1), imagesc(log(abs(fftshift(im_fft)))), axis image, colormap jet
```

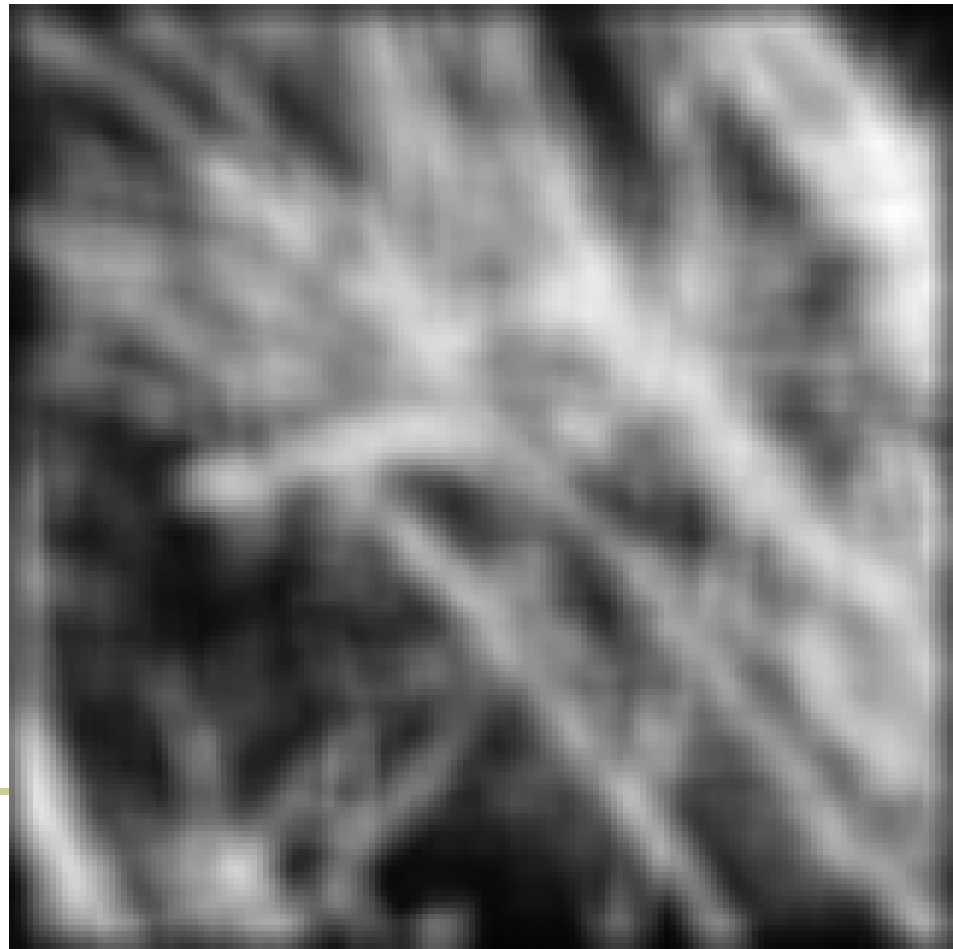
Filtering

Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?

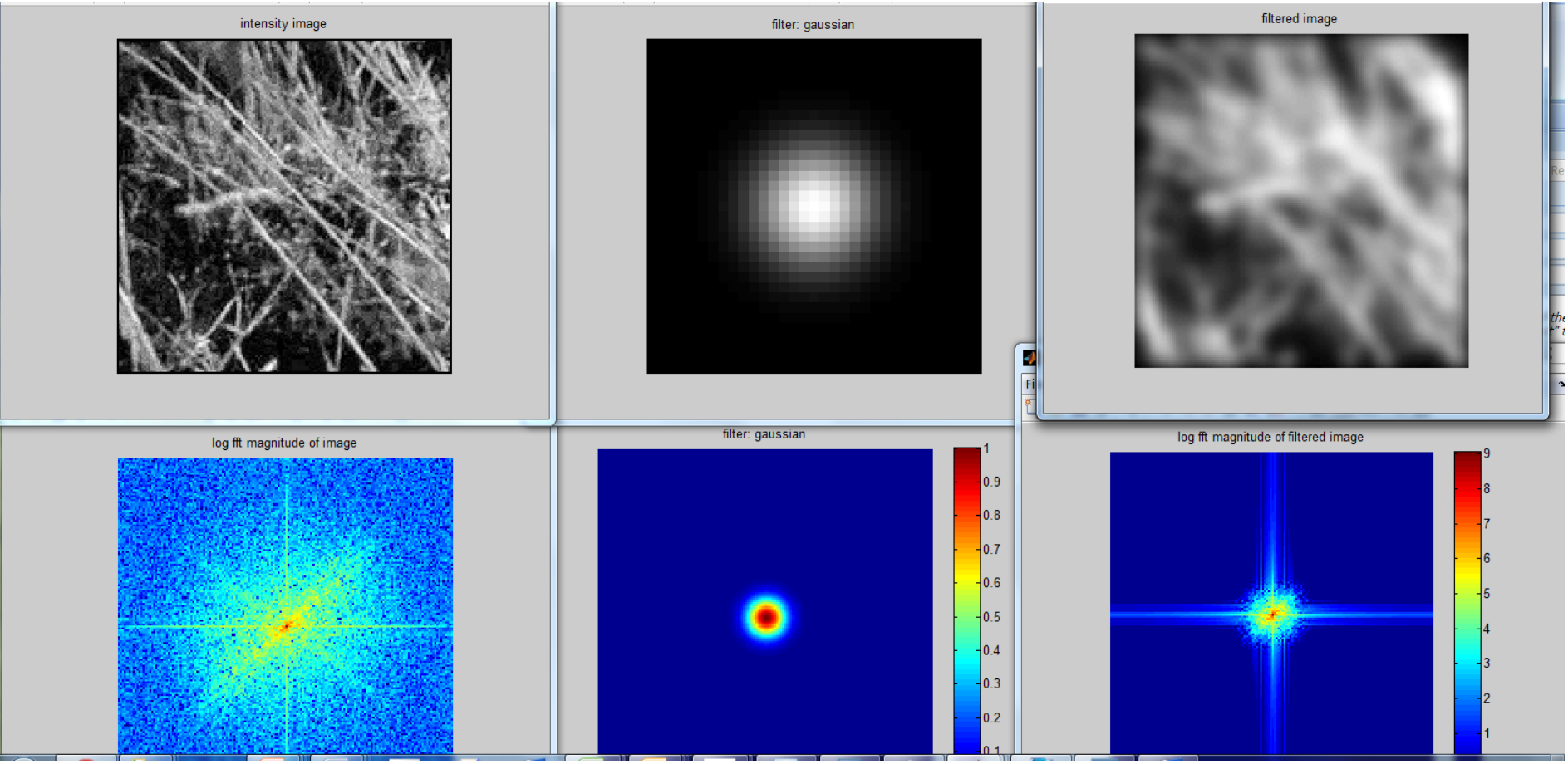
Gaussian 

Box filter 

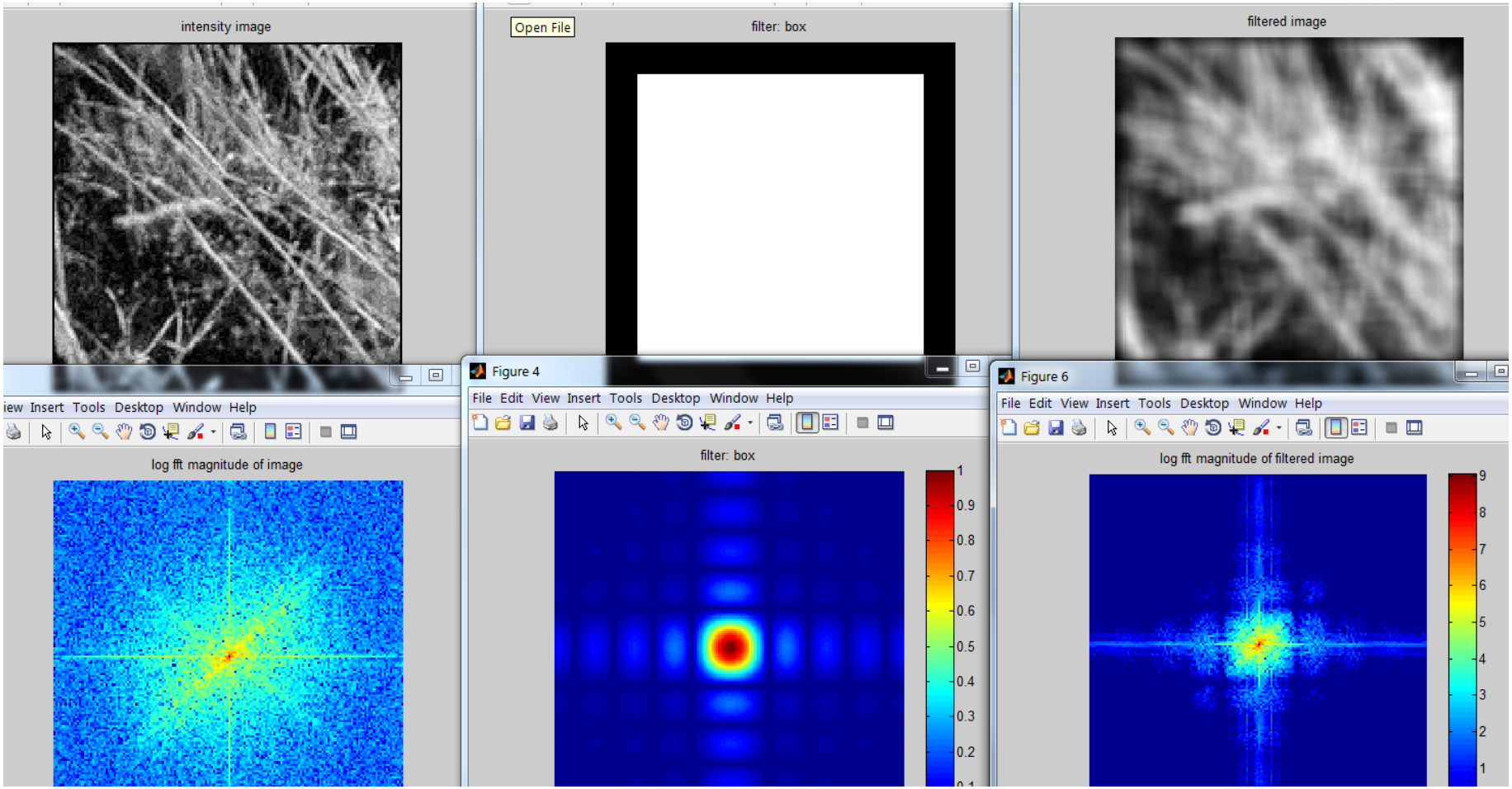




Gaussian Filter



Box Filter


intensity image
Open File
filter: box
filtered image
Figure 4
File Edit View Insert Tools Desktop Window Help
Figure 6
File Edit View Insert Tools Desktop Window Help
log fft magnitude of image
filter: box
log fft magnitude of filtered image
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
9
8
7
6
5
4
3
2
1

Today's Class

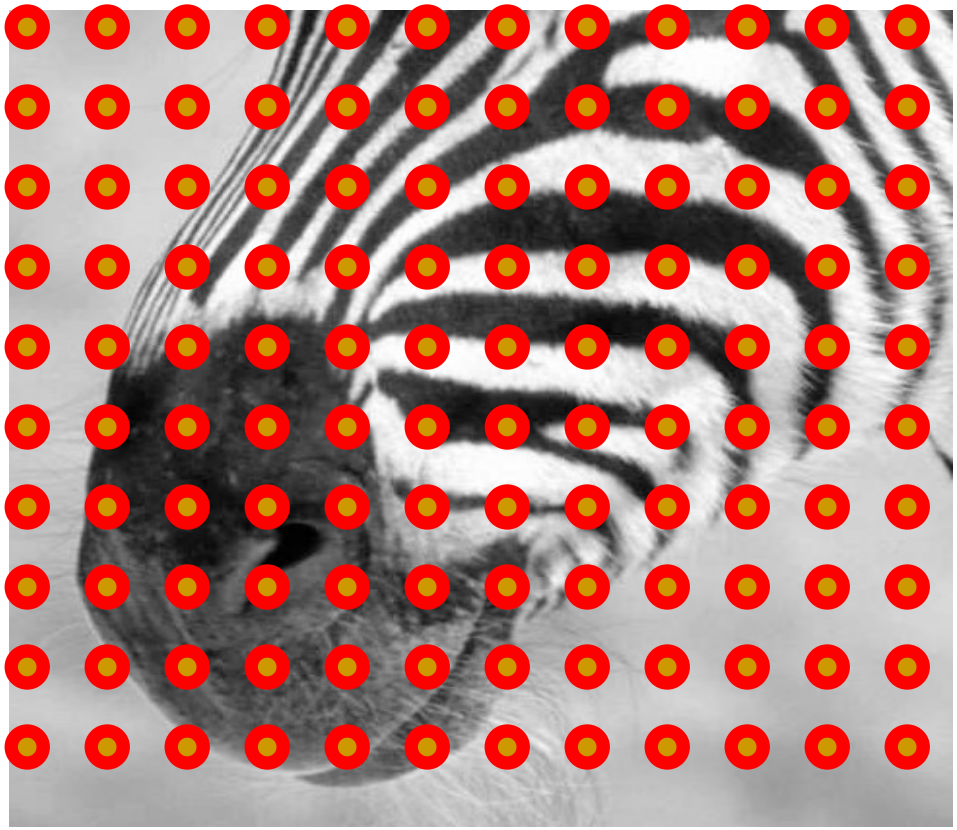
- Frequency domain and Fourier transform
- 1D Discrete Fourier Transform
- 2D Image Fourier Transform
- **Sampling**
- Hybrid Image

Sampling

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

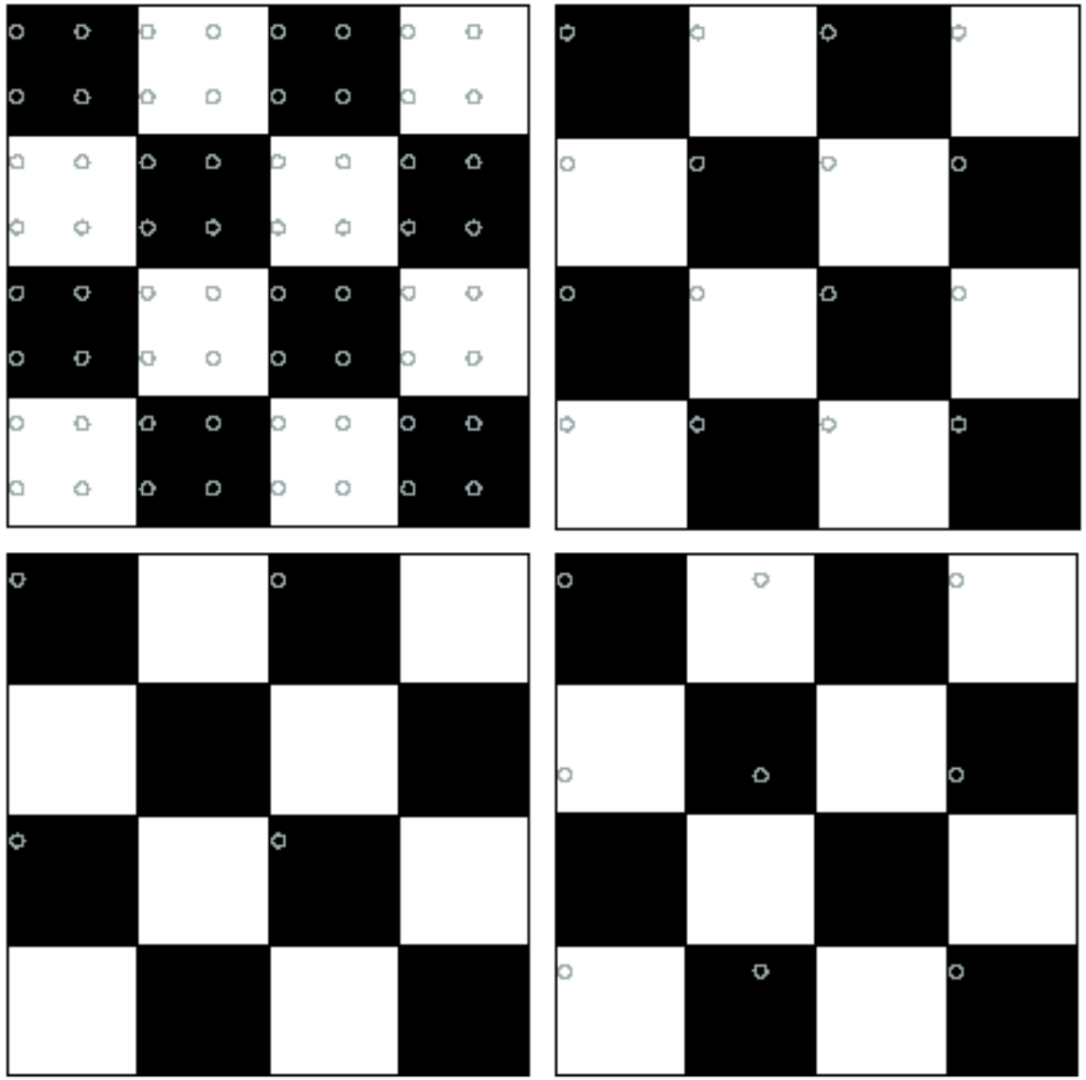
Image: http://www.flickr.com/photos/igorms/136916757/

Subsampling by a factor of 2



Throw away every other row and column
to create a 1/2 size image

How should we go about sampling



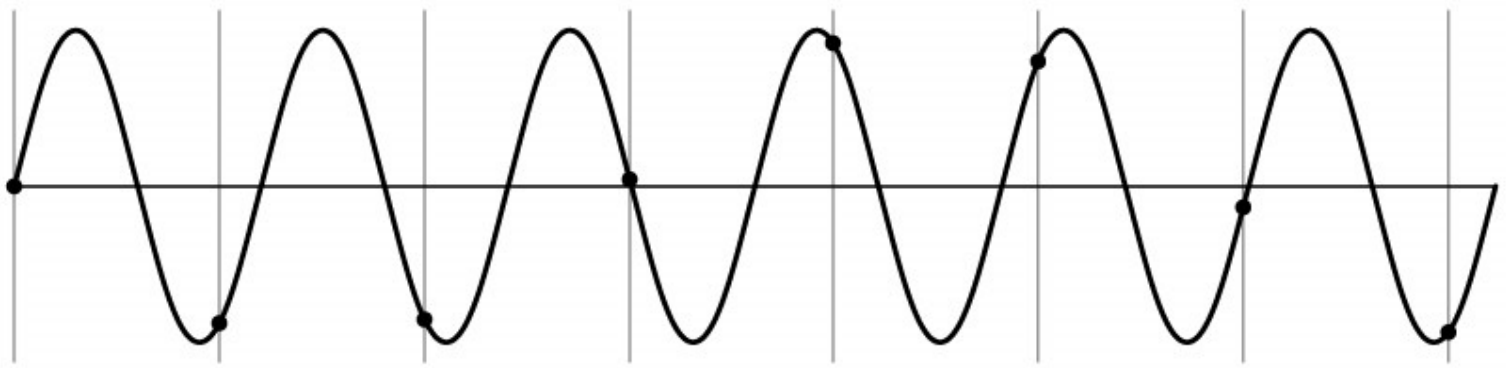
Let's resample the checkerboard by taking one sample at each circle.

In the top left board, the new representation is reasonable. Top right also yields a reasonable representation.

Bottom left is all black (dubious) and bottom right has checks that are too big.

How should we go about sampling

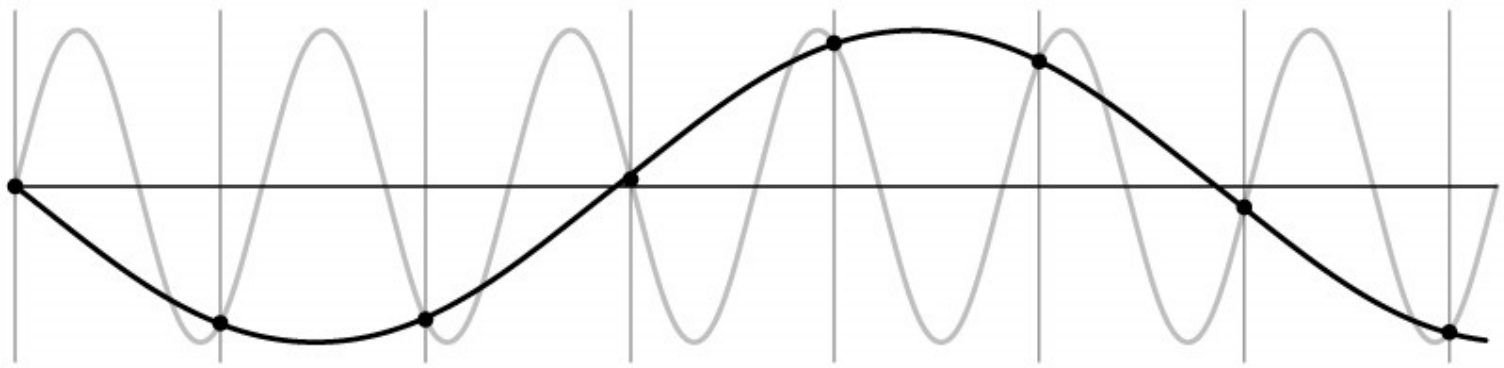
- 1D example (sinewave):



Source: S. Marschner

How should we go about sampling

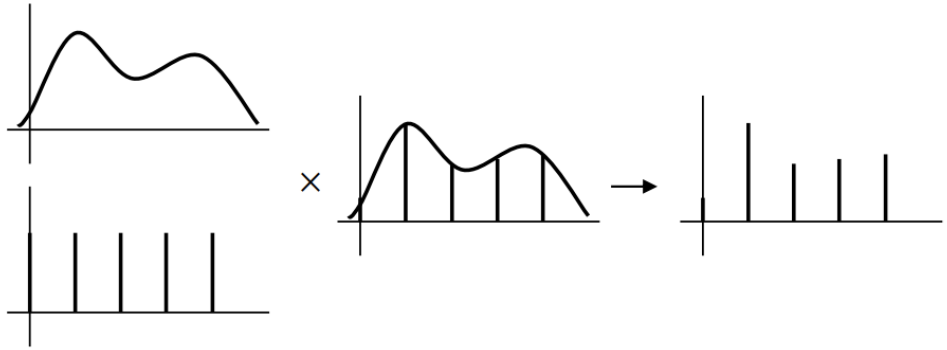
- 1D example (sinewave):



Source: S. Marschner

Fourier Interpretation: Sampling

- Sampling in the spatial domain is like multiplying with a spike function.

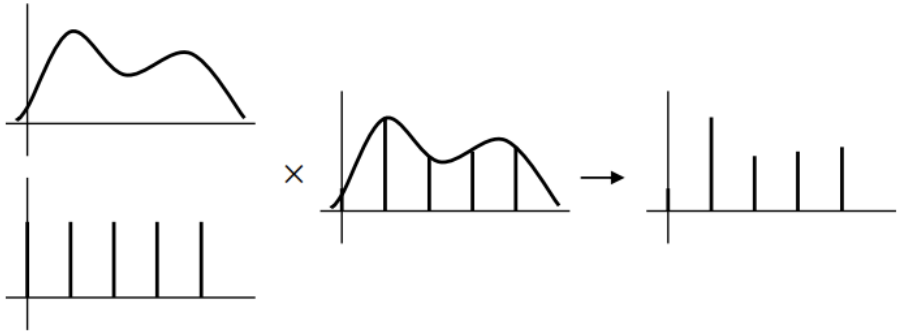


- Sampling in the frequency domain is like...

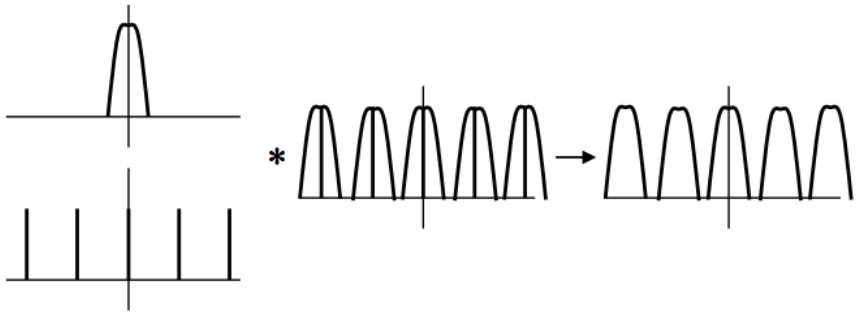
?

Fourier Interpretation: Sampling

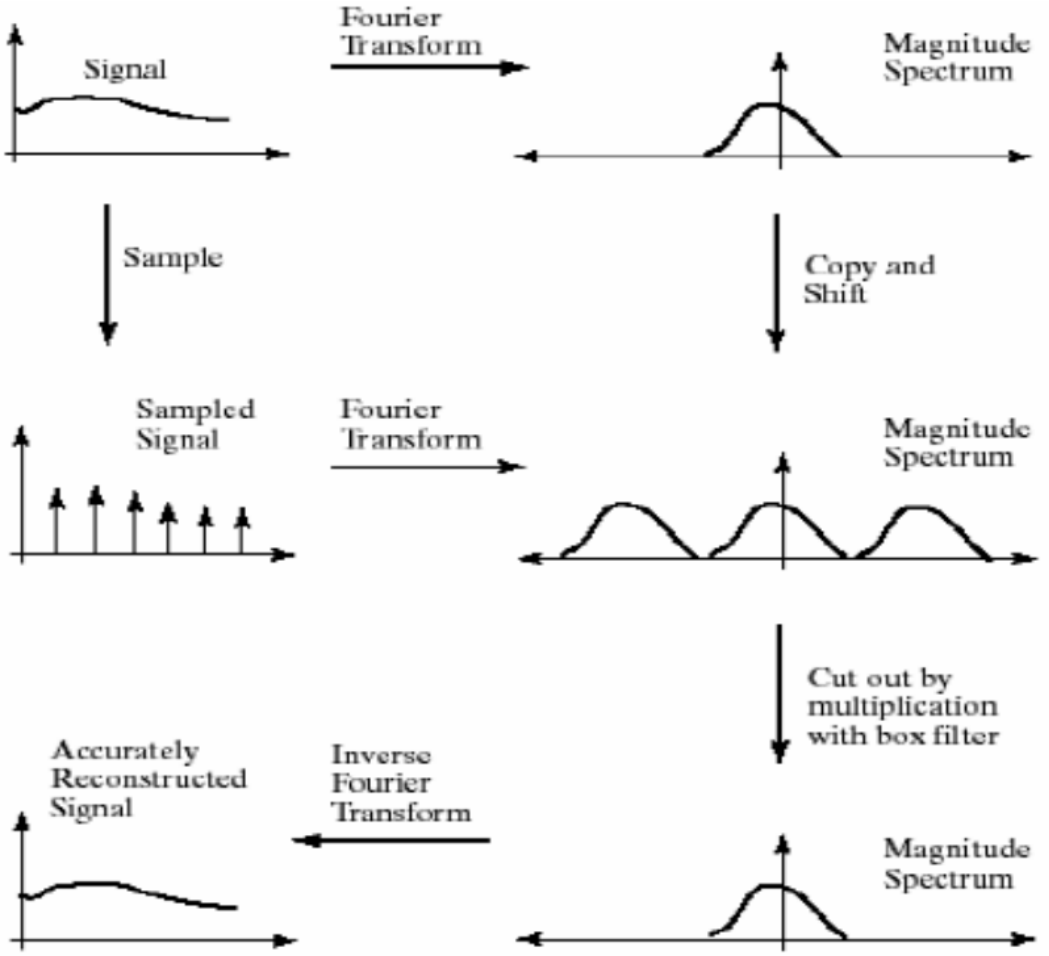
- Sampling in the spatial domain is like multiplying with a spike function.



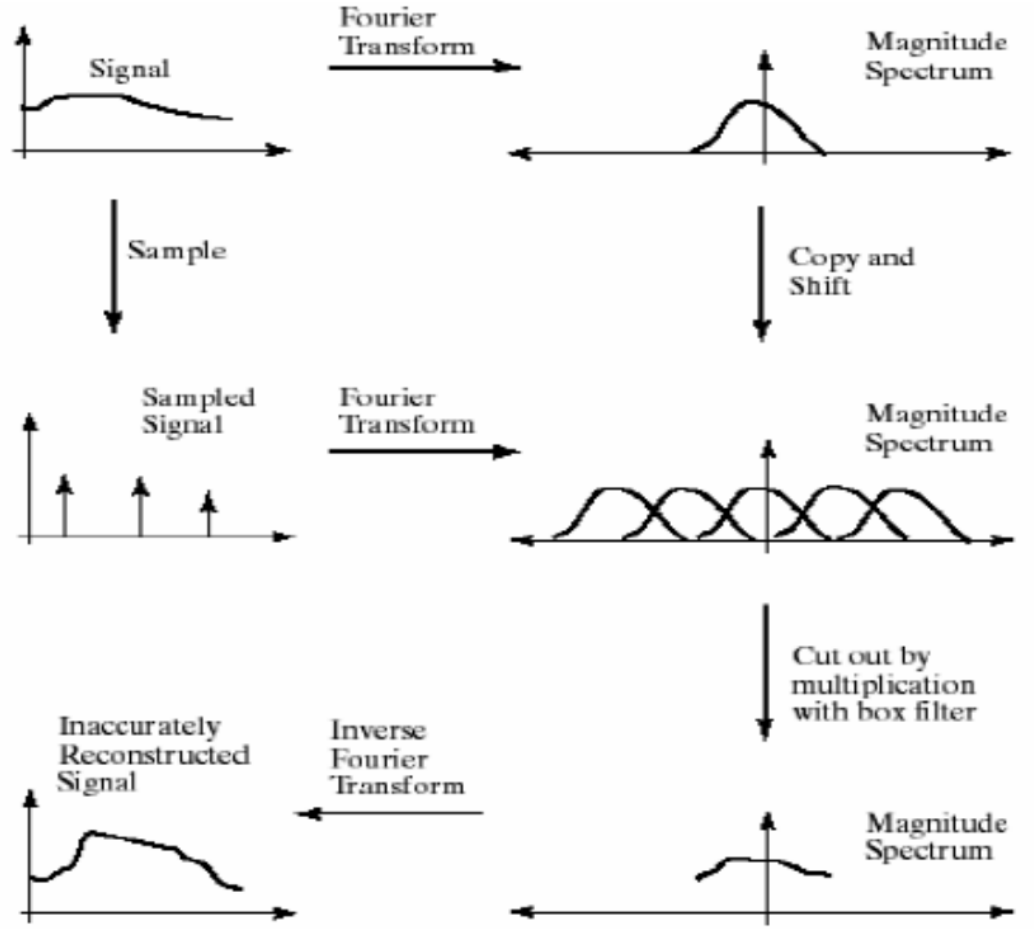
- Sampling in the frequency domain is like convolving with a spike function.



Fourier Interpretation: Sampling



Fourier Interpretation: Sampling

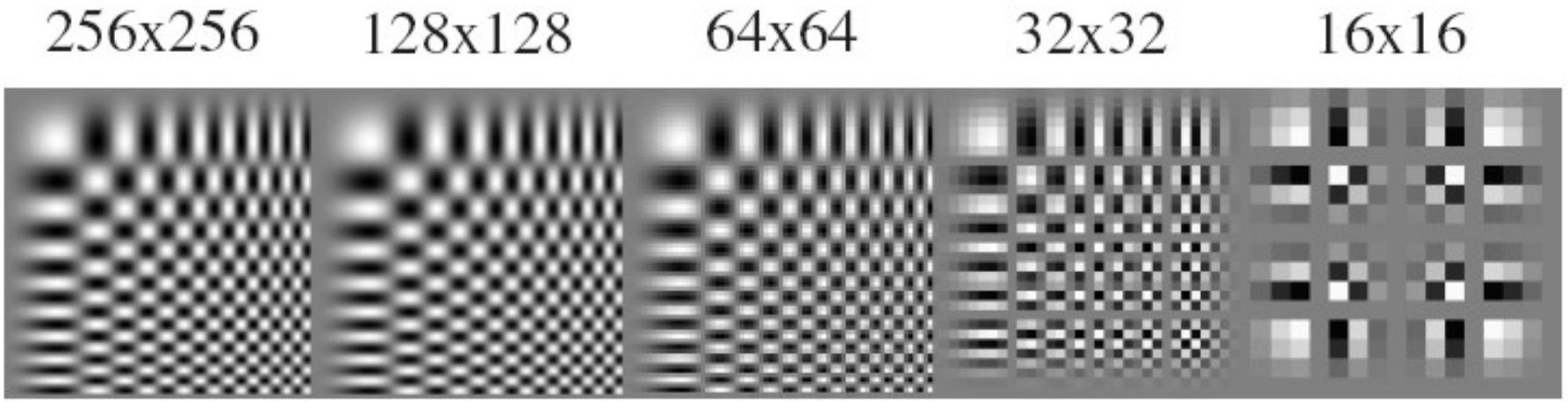


Aliasing problem

- Sub-sampling may be dangerous….
- Characteristic errors may appear:
  - “Wagon wheels rolling the wrong way in movies”
  - “Checkerboards disintegrate in ray tracing”
  - “Striped shirts look funny on color television”

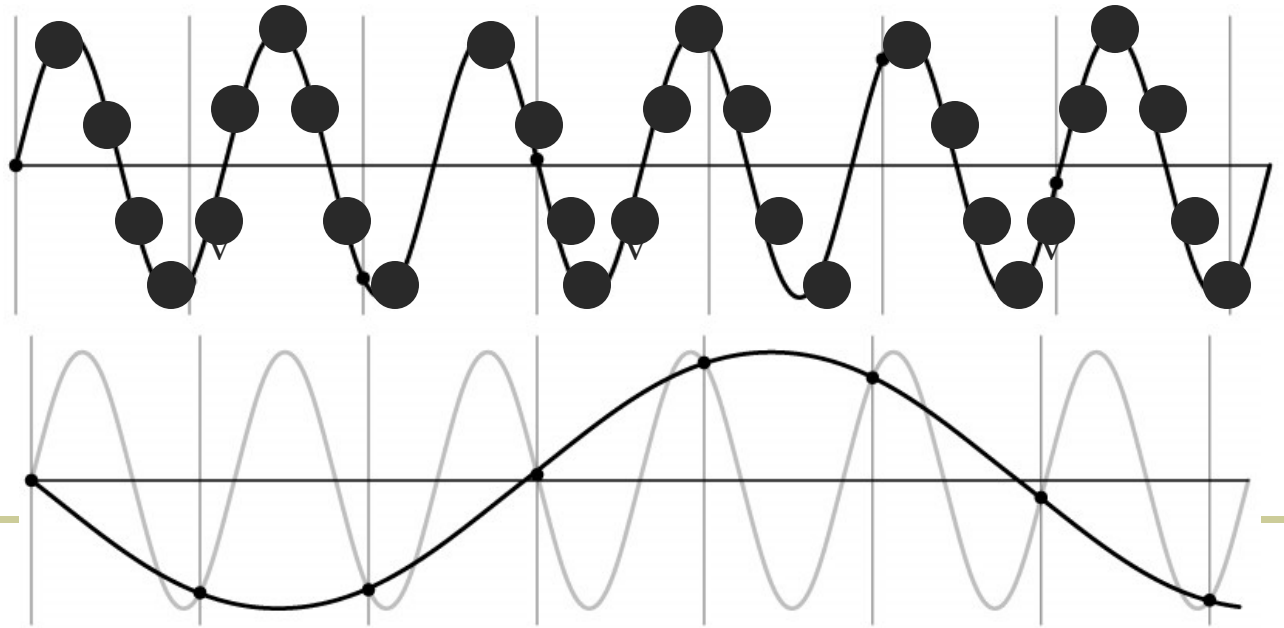
Source: D. Forsyth

Sampling and aliasing



Nyquist-Shannon Sampling Theorem

- When sampling a signal at discrete intervals, the sampling frequency must be $\geq 2 \times f_{max}$
- f_{max} = max frequency of the input signal
- This will allows to reconstruct the original perfectly from the sampled version



Anti-aliasing

Solutions:

- Sample more often

- Get rid of all frequencies that are greater than half the new sampling frequency
  - Will lose information
  - But it's better than aliasing
  - Apply a smoothing filter

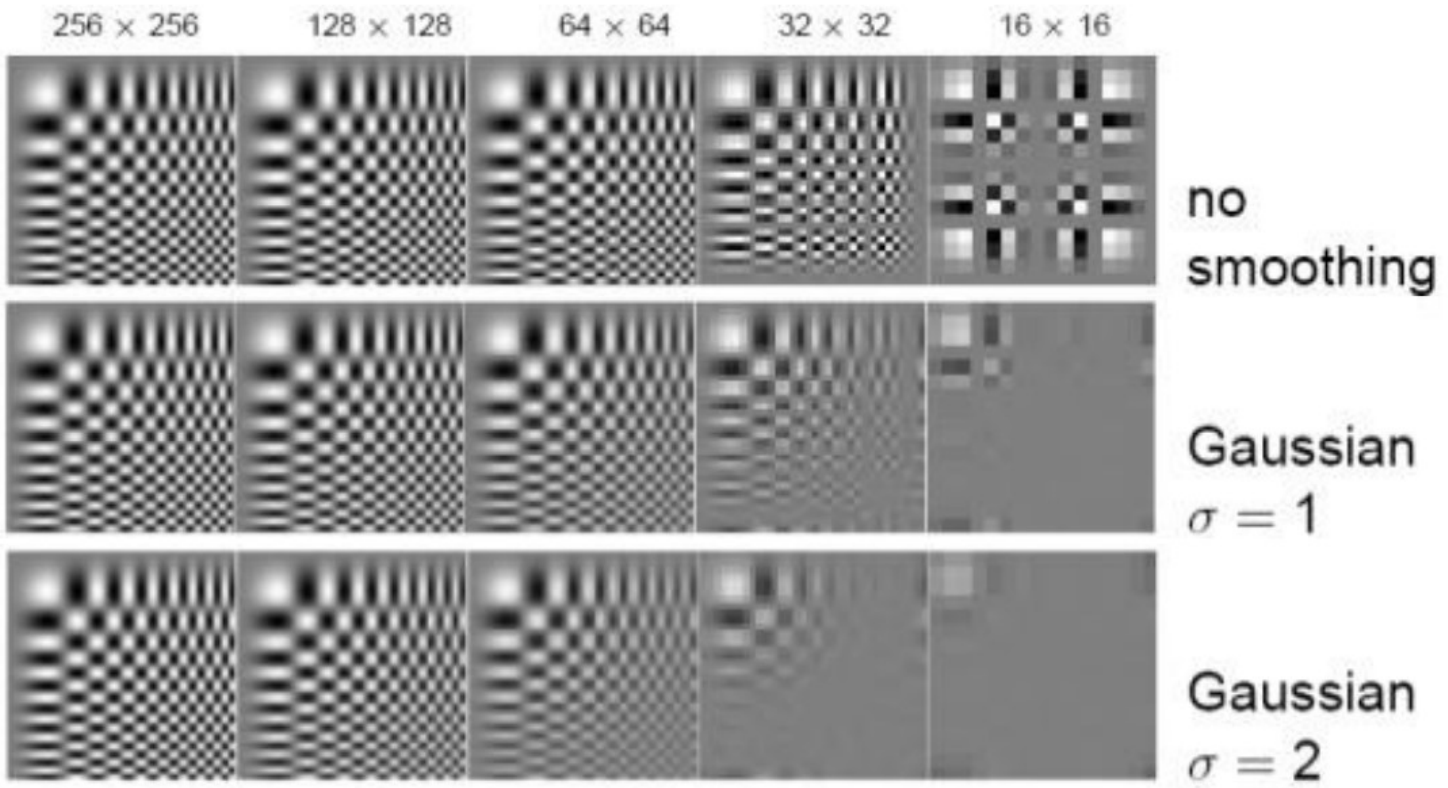
Algorithm for downsampling by factor of 2

1. Start with image(h, w)
2. Apply low-pass filter

   im_blur = imfilter(image, fspecial('gaussian', 7, 1))
3. Sample every other pixel

   im_small = im_blur(1:2:end, 1:2:end);

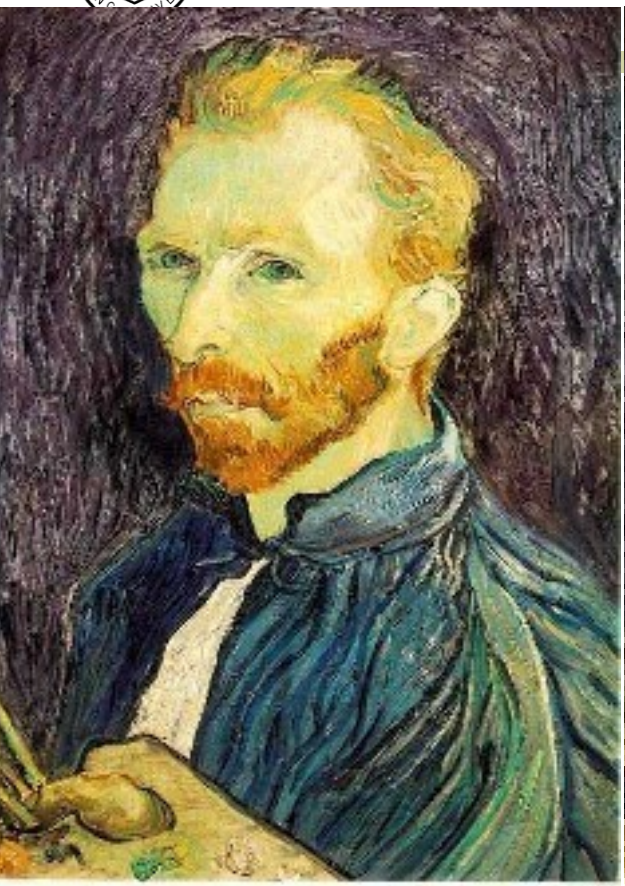
Anti-aliasing



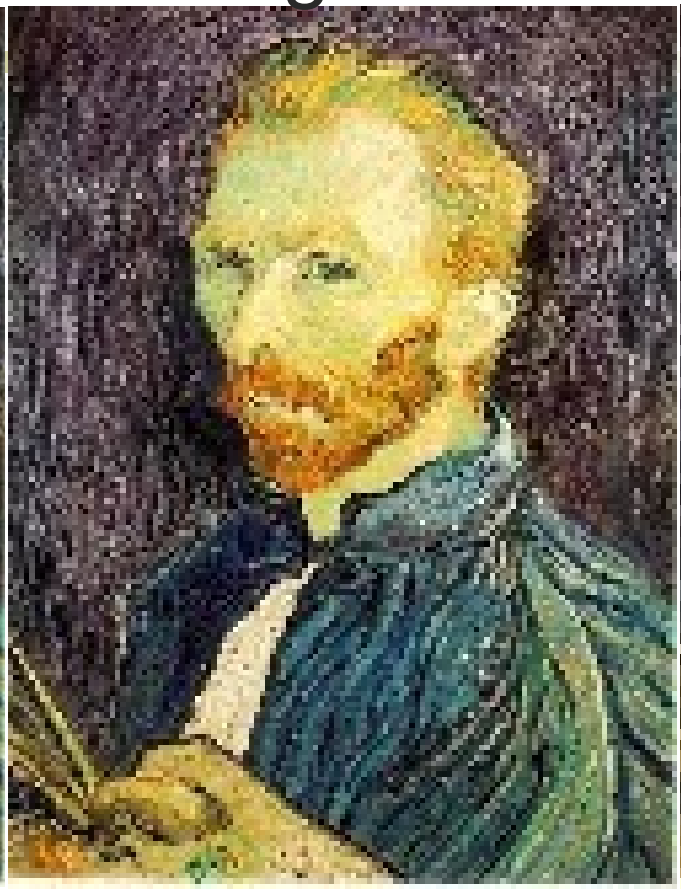
Note: We cannot recover the high frequencies, but we can avoid artifacts by smoothing before resampling.

Forsyth and Ponce 2002

Subsampling without pre-filtering



1/2

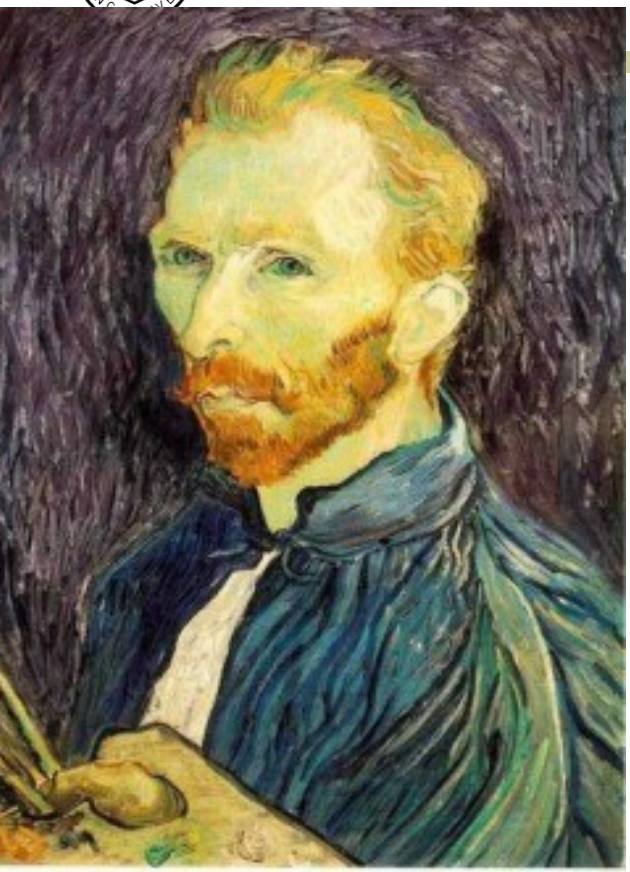


1/4 (2x zoom)

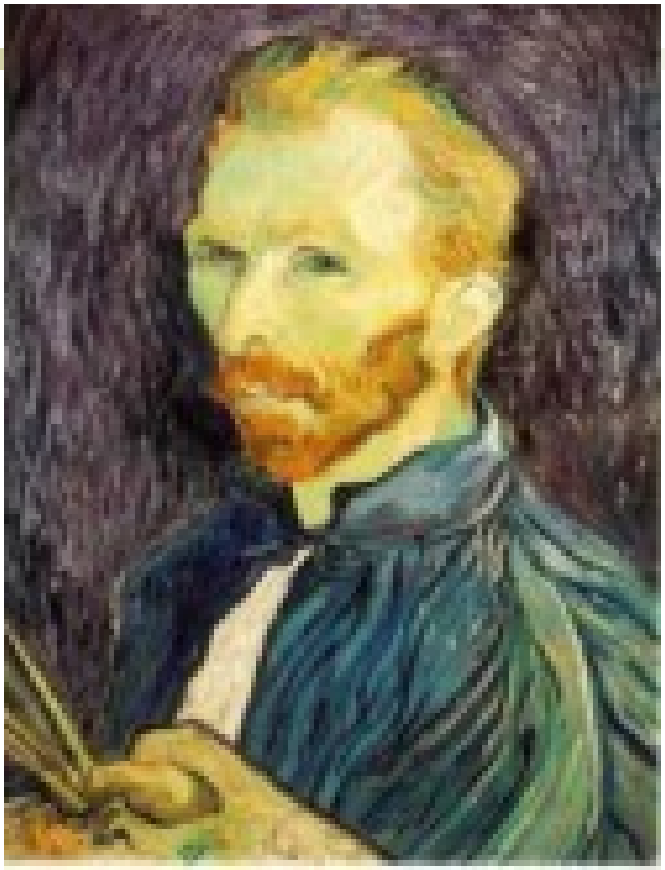
1/8 (4x zoom)

Slide by Steve Seitz

Subsampling with Gaussian pre-filtering



Gaussian 1/2



G 1/4

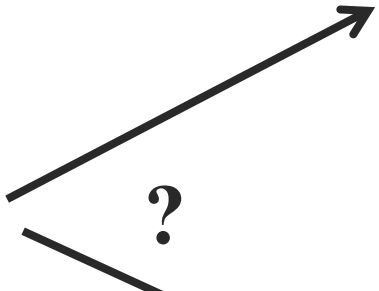
G 1/8

Slide by Steve Seitz

Today's Class

- Frequency domain and Fourier transform
- 1D Discrete Fourier Transform
- 2D Image Fourier Transform
- Sampling
- **Hybrid Image**

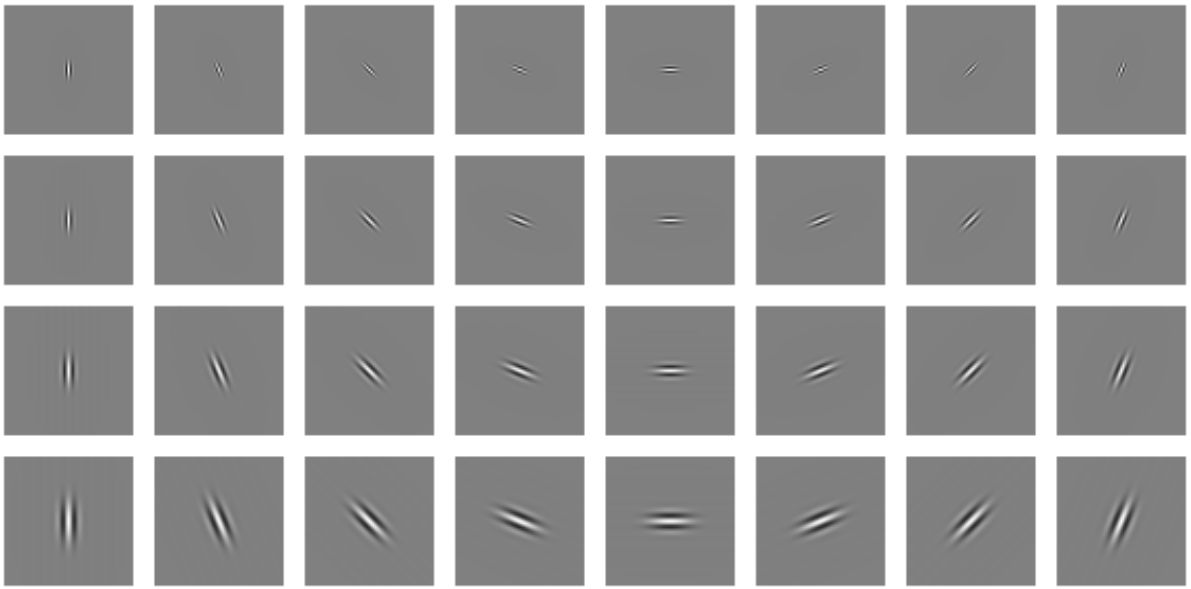
Why do we get different, distance-dependent interpretations of hybrid images?



?

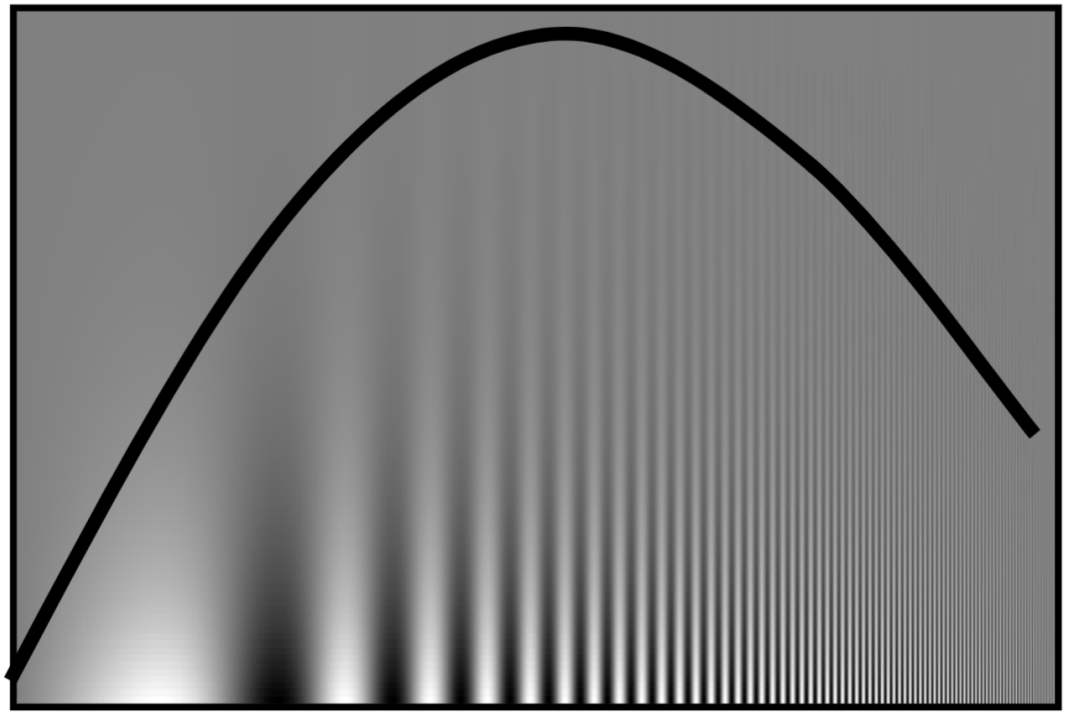
Clues from Human Perception

- Early processing in humans filters for various orientations and scales of frequency
- Perceptual cues in the mid-high frequencies dominate perception
- When we see an image from far away, we are effectively subsampling it



Early Visual Processing: Multi-scale edge and blob filters

Frequency Domain and Perception

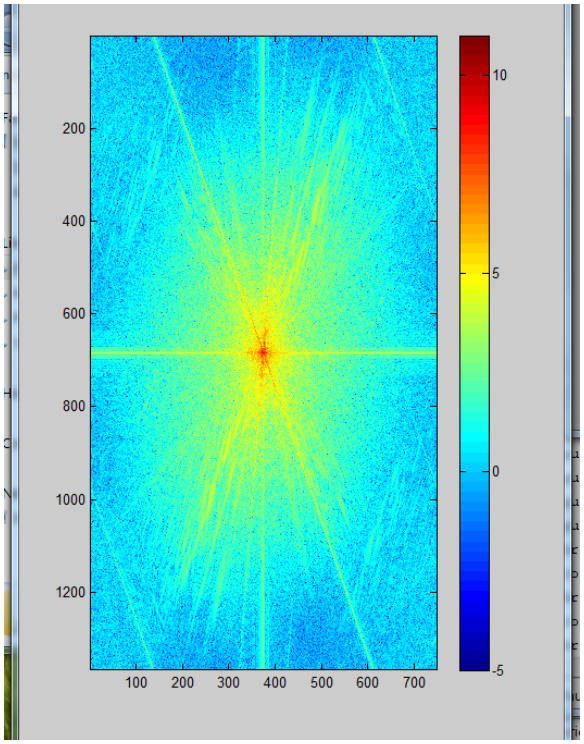


Campbell-Robson contrast sensitivity curve

slide: A. Efros

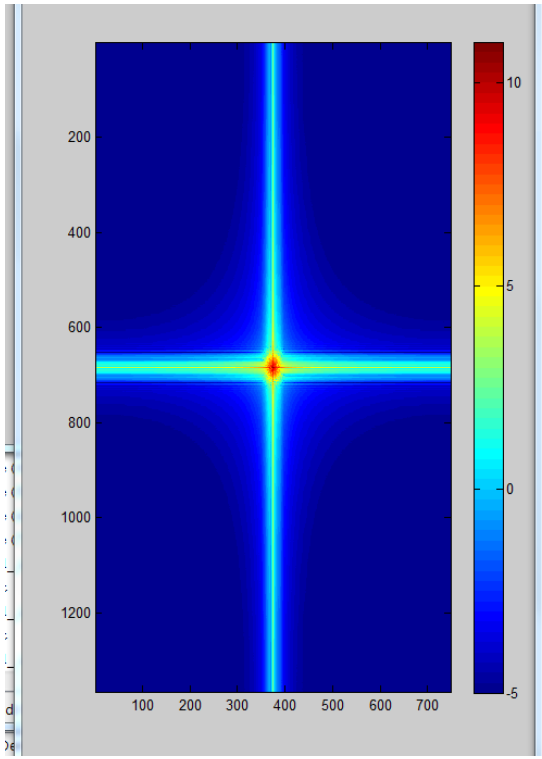
Hybrid Image in FFT

Hybrid Image



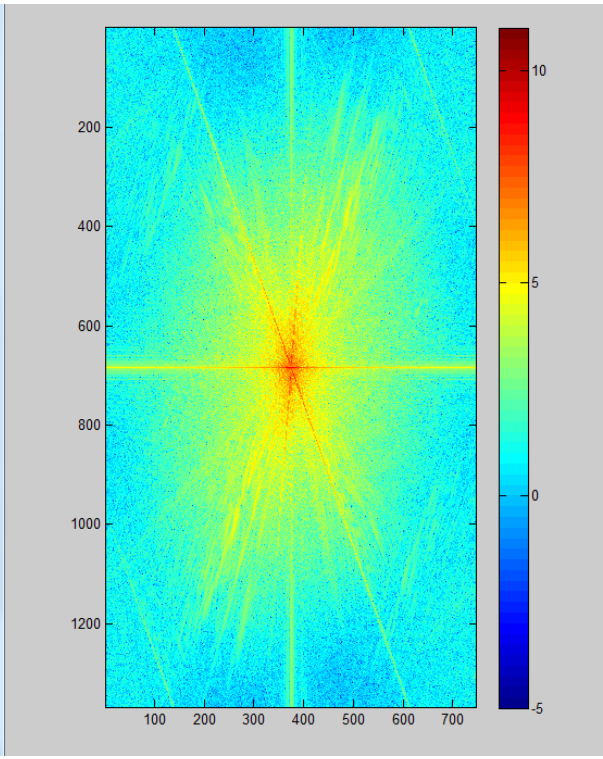
=

Low-passed Image

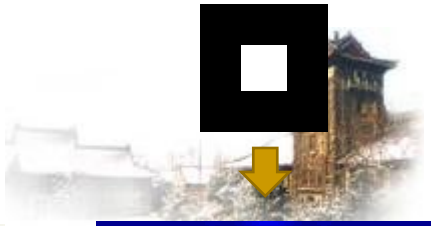


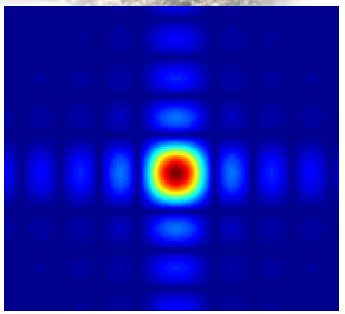
+

High-passed Image



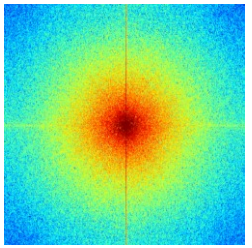
Things to Remember





- Sometimes it makes sense to think of images and filtering in the frequency domain
  - Fourier analysis

- Can be faster to filter using FFT for large images (N logN vs. N^2 for auto-correlation)



- Remember to low-pass before sampling