

Image Filtering (Frequency Domain)

Dr. Sander Ali Khowaja,

Assistant Professor, Department of Telecommunication Engineering
Faculty of Engineering and Technology, University of Sindh, Pakistan

Senior Member, IEEE – Member, ACM

<https://sander-ali.github.io>

Computer Vision & Image Processing



Some AI related news

Augmented reality headset enables users to see hidden objects

The device could help workers locate objects for fulfilling e-commerce orders or identify parts for assembling products.

 Watch Video

Adam Zewe | MIT News Office

February 27, 2023



- Linear filtering is sum of dot product at each position
 - Can smooth, sharpen, translate (among many other uses)
- Gaussian filters
 - Low pass filters, separability, variance
- Attend to details:
 - filter size, extrapolation, cropping
- Applications:
 - Noise models and nonlinear image filters
 - Texture representation



$$\frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$



Today's class

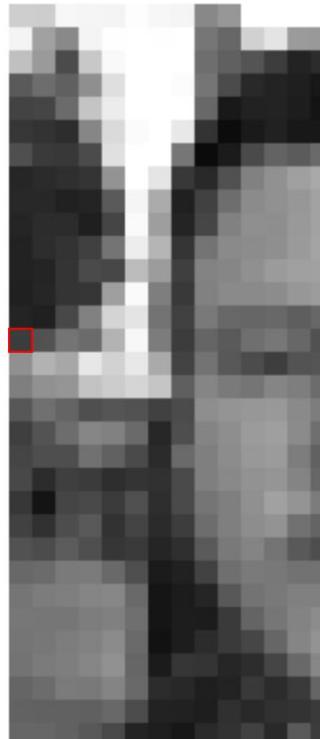
- Review of image filtering in spatial domain
- Fourier transform and frequency domain
- Frequency view of filtering
- Image downsizing and interpolation
- Goals:
 - Understand 2D Fourier transform
 - Understand how to implement filtering in Fourier domain
 - Understand aliasing and how to prevent aliasing



Application: Photometric Stereo

- <http://setosa.io/ev/image-kernels/>

```
208 225 247 248 244 253 247 245 131 151 255 255 255 255 234 207 231 251 254 254 254 252 255 255 254 255 247  
244 181 137 244 254 255 254 255 110 103 209 228 155 153 238 193 74 52 68 173 255 254 254 255 255 254 253 244 184  
192 154 75 200 249 255 255 255 110 98 84 81 35 44 89 53 44 45 43 54 140 213 233 255 255 254 255 254 253 244 184  
93 109 98 143 223 255 255 255 117 75 41 35 31 24 25 38 45 44 44 44 46 81 118 148 234 252 254 255 248 231 248 255 254  
87 89 107 198 238 255 255 104 25 34 35 29 20 25 34 32 30 32 34 53 85 100 142 231 242 247 249 255 255 255 254  
55 51 45 134 218 251 255 255 232 51 12 28 33 24 24 46 75 82 78 71 68 58 53 67 90 138 228 238 158 253 248 249 255  
79 58 58 75 224 255 255 118 11 27 74 91 91 108 140 162 173 173 173 172 158 137 92 46 78 187 217 208 254 222 233 255  
38 43 52 147 255 229 56 41 81 129 145 160 169 169 169 172 178 179 177 177 172 160 31 82 209 238 259 244 249 255  
40 40 33 38 90 245 171 32 85 110 139 145 151 162 171 174 178 179 182 184 187 183 173 182 71 45 167 255 254 255 254 255  
37 44 44 31 89 251 158 38 70 128 143 143 153 162 171 175 177 178 182 191 194 188 180 170 120 51 137 255 250 254 255  
34 45 51 64 118 237 181 53 178 138 140 143 154 184 178 178 174 177 183 188 185 185 183 178 140 69 141 254 252 225 249 255  
34 38 52 74 71 188 158 63 131 134 144 155 160 161 173 173 179 189 193 193 195 187 182 158 93 148 250 254 214 247 255  
32 54 54 159 250 128 57 129 138 138 140 151 158 168 168 171 178 180 187 188 185 185 183 180 122 138 242 259 255 254 254  
36 52 72 129 212 228 115 85 121 104 102 104 94 103 134 158 170 162 125 108 121 143 155 190 191 104 134 230 253 255 255 251  
81 92 118 107 179 247 124 80 101 90 115 119 103 81 94 147 191 178 128 98 123 153 147 181 200 92 100 222 207 187 227 215  
144 178 187 231 210 232 170 67 115 88 78 82 83 85 88 139 192 190 135 80 53 99 141 185 201 97 79 152 245 235 248 249  
127 149 195 204 213 197 95 133 122 117 133 128 138 110 139 191 197 187 129 127 148 147 171 188 110 121 228 233 180 215 212  
87 152 100 79 85 92 85 75 142 148 151 153 138 125 120 149 191 180 193 175 174 193 198 190 208 121 239 219 149 198 195  
83 109 134 129 103 39 78 132 142 155 159 139 111 124 184 195 200 188 192 191 195 200 202 200 143 217 253 249 242 238 234  
89 78 78 183 97 74 43 152 127 140 152 155 97 121 160 165 154 174 174 183 198 198 202 208 209 169 247 254 255 254 254  
72 44 83 59 48 52 49 74 127 137 148 149 132 103 79 90 134 141 188 161 160 207 204 203 218 193 236 244 251 242 238 243  
55 20 69 73 59 80 48 74 117 127 144 161 148 124 125 120 158 187 193 182 189 208 201 205 214 194 194 185 197 188 193 193  
85 49 77 89 50 88 43 81 109 127 141 147 173 100 121 145 148 169 181 178 181 201 201 205 202 174 188 189 178 183 188 184  
82 78 92 79 54 58 37 47 90 121 132 116 89 79 111 148 183 149 122 134 180 197 198 178 149 148 152 151 157 159 188  
104 107 122 123 109 79 27 33 66 111 132 120 104 147 175 130 198 163 181 171 200 187 189 158 148 146 139 137 141 140 145  
117 134 124 127 133 135 105 21 28 37 88 115 121 128 141 142 168 202 212 153 184 186 180 188 154 148 144 140 151 151 147 144  
119 118 118 125 128 111 21 29 28 58 100 110 131 140 151 159 188 201 205 192 185 188 149 168 119 144 147 143 140 141 144 148  
117 119 125 130 139 108 18 29 44 58 70 102 133 147 168 197 212 215 210 195 177 152 133 125 57 59 128 151 145 142 141 141  
115 123 128 134 145 102 27 54 52 38 45 69 105 125 175 189 193 216 208 188 130 111 184 203 74 5 121 151 142 142 143 148  
101 108 123 121 132 105 44 40 31 35 57 44 58 101 147 144 138 183 145 94 90 145 198 187 94 48 165 180 142 144 142 145  
98 97 97 98 104 78 34 33 30 48 41 49 51 58 74 53 55 68 83 89 150 188 209 156 82 108 140 149 125 133 131 131 131  
102 102 97 88 73 35 30 23 42 50 85 41 90 80 59 51 57 82 123 157 187 205 169 82 98 151 105 101 154 135 130 129
```

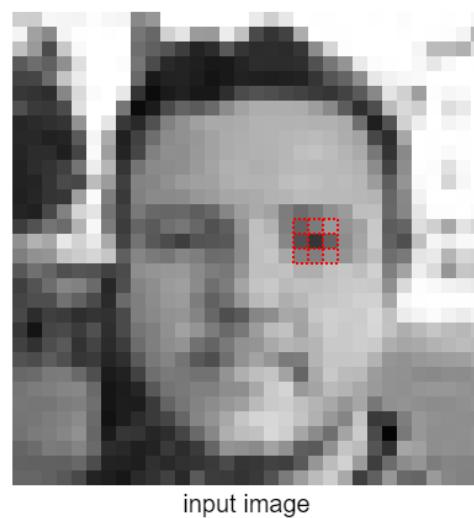


Let's walk through applying the following 3x3 **sharpen** kernel to the image of a face from above.

sharpen ▾

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Below, for each 3x3 block of pixels in the image on the left, we multiply each pixel by the corresponding entry of the kernel and then take the sum. That sum becomes a new pixel in the image on the right. Hover over a pixel on either image to see how its value is computed.



$$\begin{aligned} & \left(\begin{array}{ccc} 98 & + & 123 & + & 153 \\ \times 0 & & \times -1 & & \times 0 \end{array} \right) \\ & + \left(\begin{array}{ccc} 80 & + & 53 & + & 99 \\ \times -1 & & \times 5 & & \times -1 \end{array} \right) \\ & + \left(\begin{array}{ccc} 129 & + & 127 & + & 148 \\ \times 0 & & \times -1 & & \times 0 \end{array} \right) \\ = & -164 \end{aligned}$$

kernel:
sharpen ▾

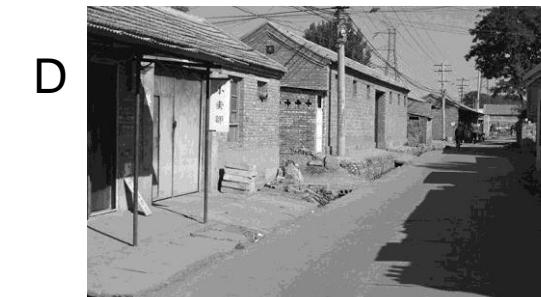
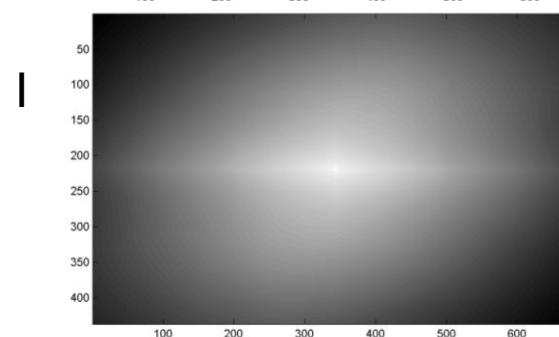
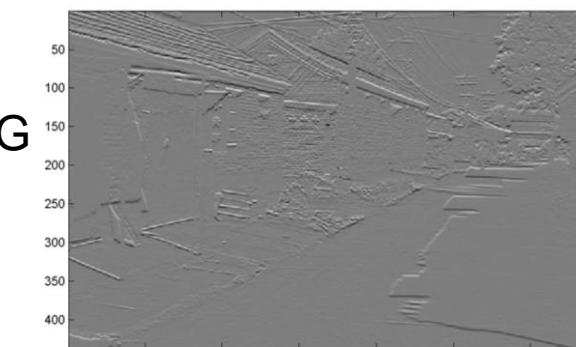
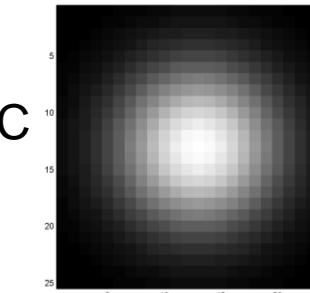
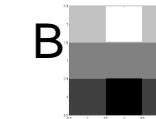
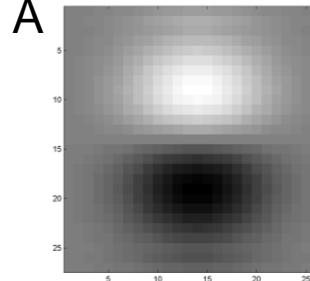


Review : Questions

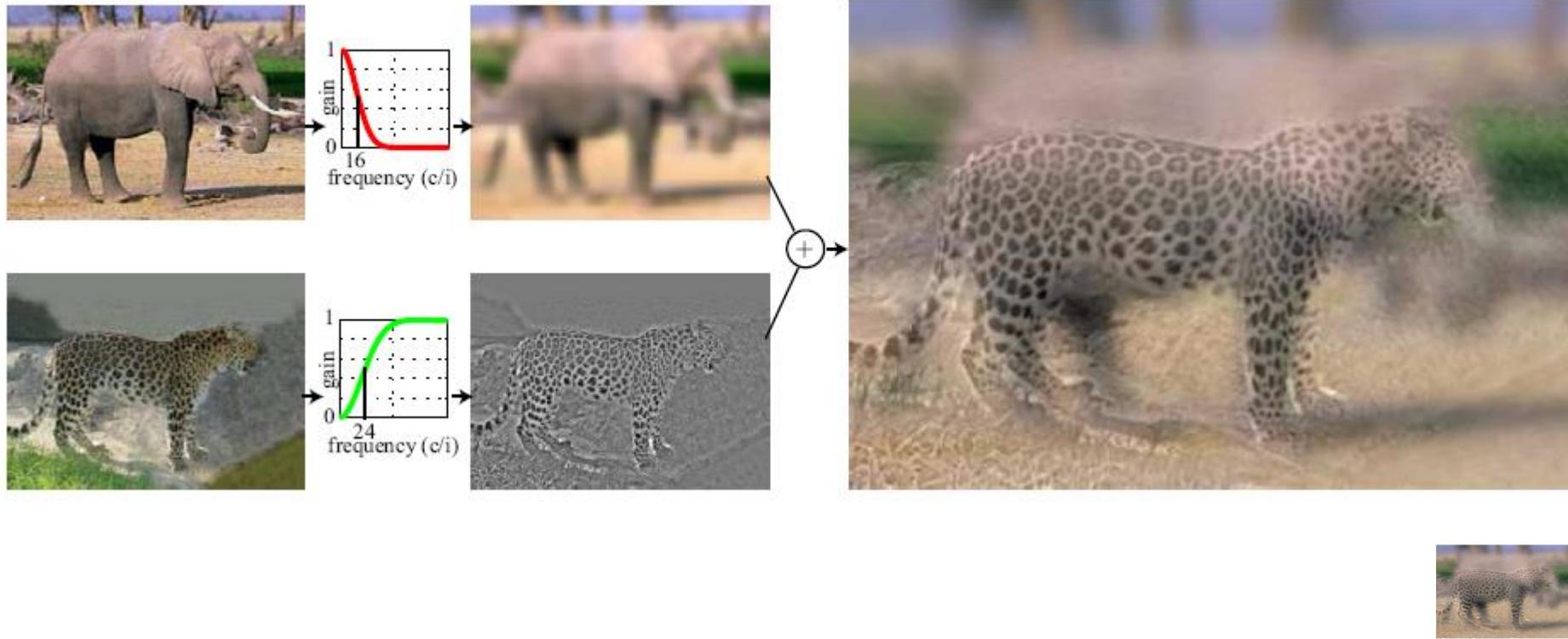
Fill in the blanks:

$$\begin{array}{l} \text{a) } \underline{\quad} = D * B \\ \text{b) } \underline{A} = \underline{\quad} * \underline{\quad} \\ \text{c) } F = D * \underline{\quad} \\ \text{d) } \underline{\quad} = D * \underline{D} \end{array}$$

Filtering Operator



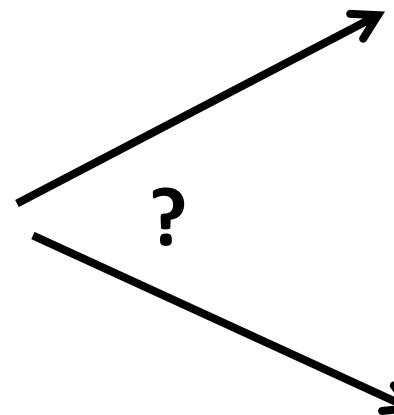
Hybrid Images



- A. Oliva, A. Torralba, P.G. Schyns,
“Hybrid Images,” SIGGRAPH 2006

Slide credit: Derek Hoiem

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



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



Slide credit: Derek Hoiem

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

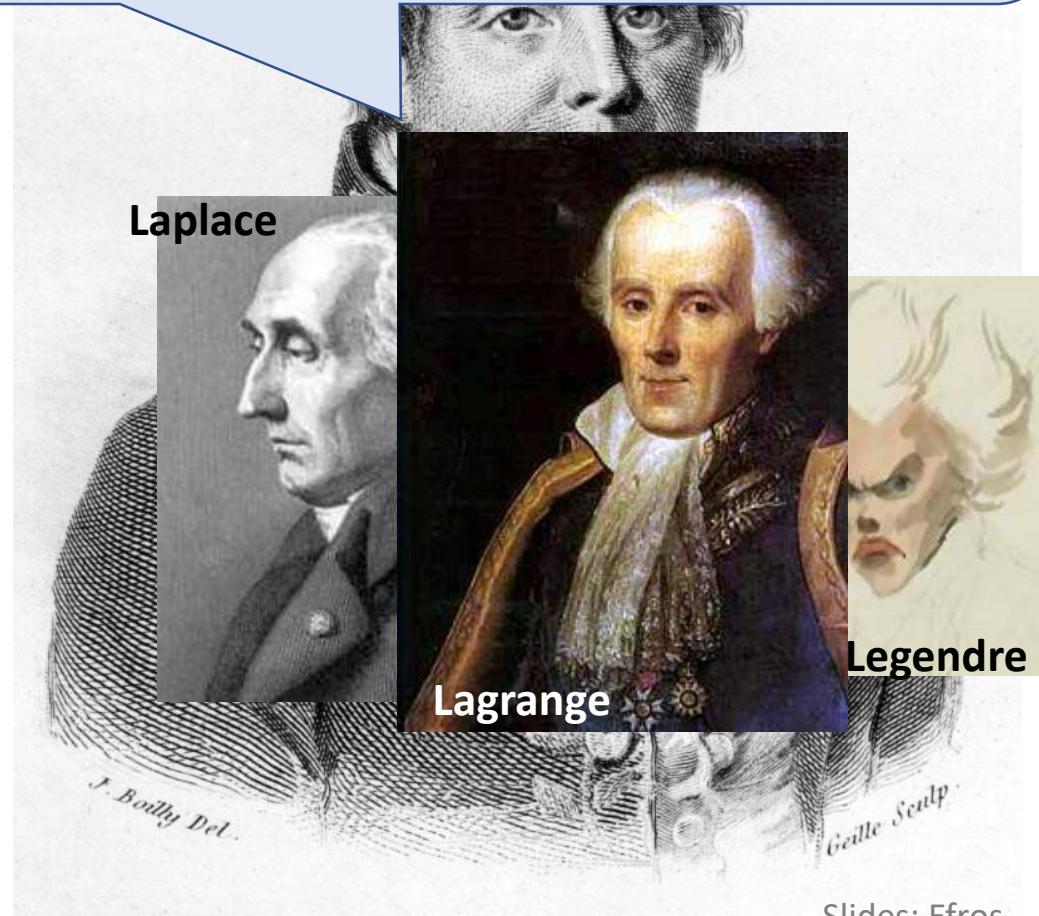
Jean Baptiste Joseph Fourier (1768-1830)

had crazy idea (1807):

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

- 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.



Fourier, Joseph

Fourier, Joseph (1768-1830)



© 1996-2007 Eric W. Weisstein

French mathematician who discovered that any periodic motion can be written as a superposition of sinusoidal and cosinusoidal vibrations. He developed a mathematical theory of [heat](#) in *Théorie Analytique de la Chaleur (Analytic Theory of Heat)*, (1822), discussing it in terms of differential equations.

Fourier was a friend and advisor of Napoleon. [Fourier believed that his health would be improved by wrapping himself up in blankets, and in this state he tripped down the stairs in his house and killed himself.](#) The paper of [Galois](#) which he had taken home to read shortly before his death was never recovered.

SEE ALSO: [Galois](#)

Add



Dr. Sander Ali Khowaja

How would math have changed if the Slanket or Snuggie had been invented?

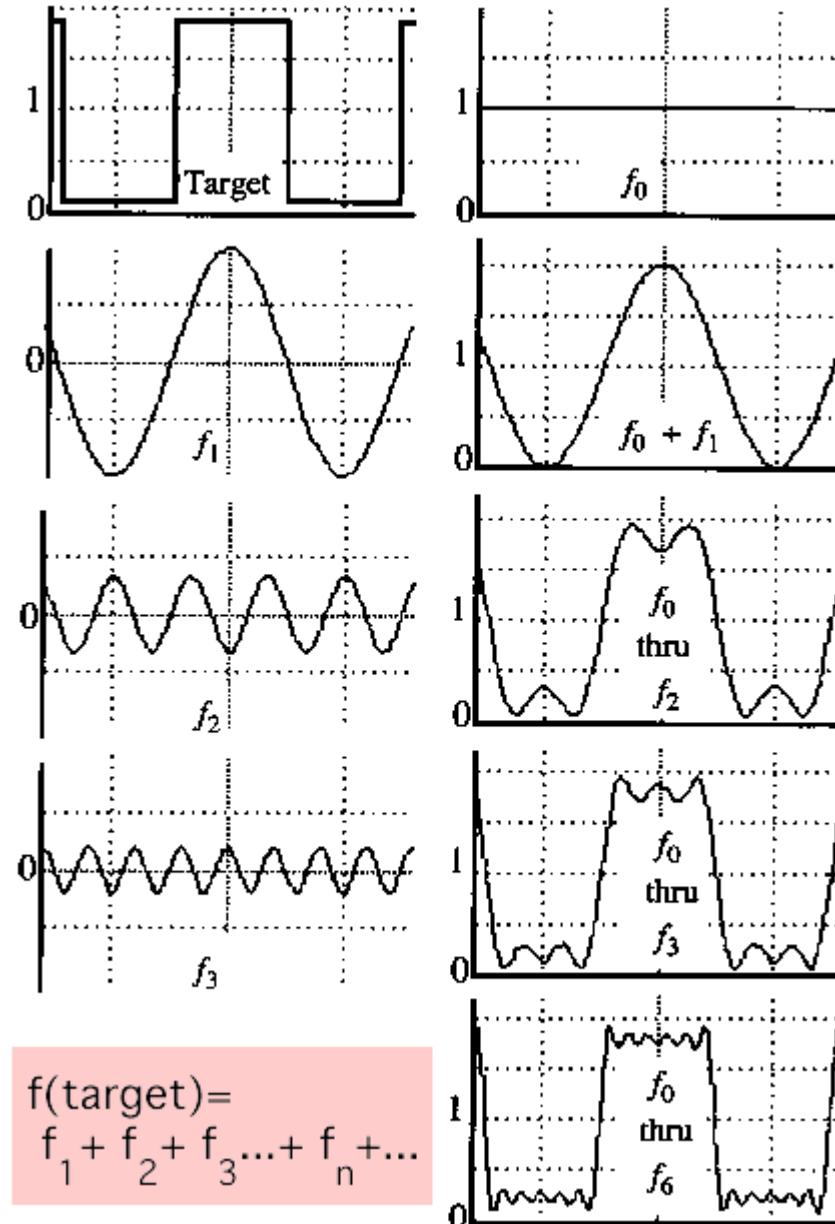


A Sum of Sines

Our building block:

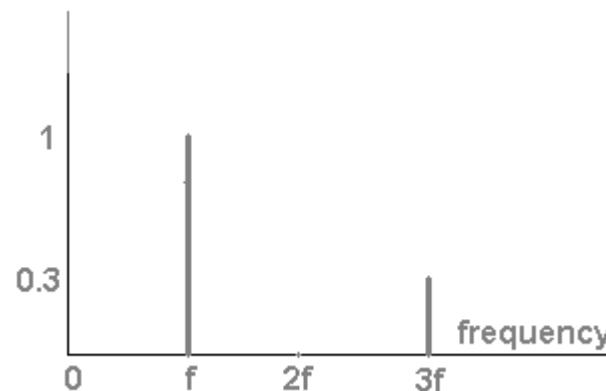
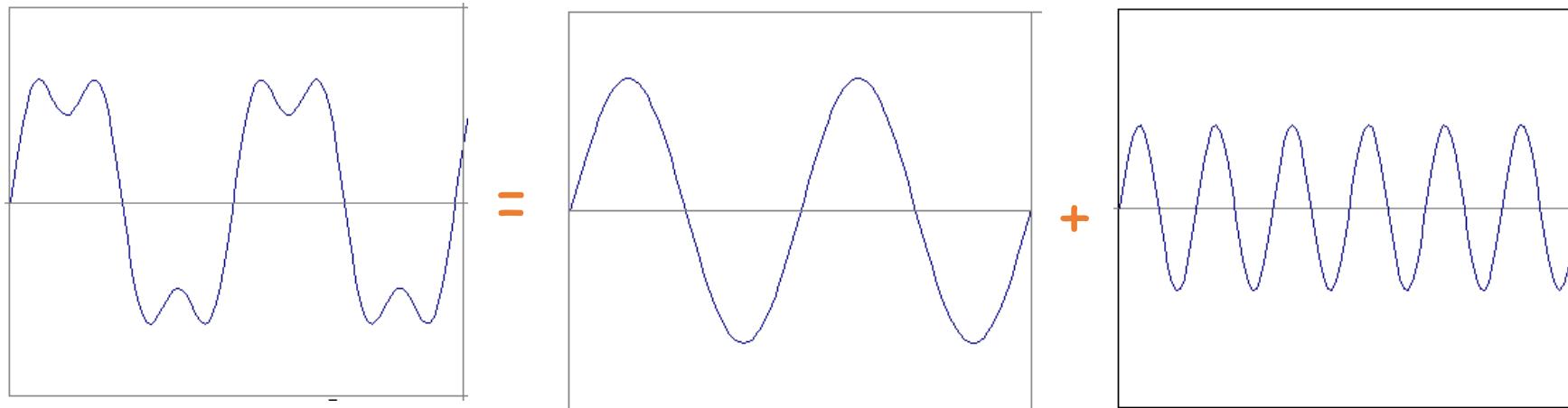
$$A \sin(\omega x + \phi)$$

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



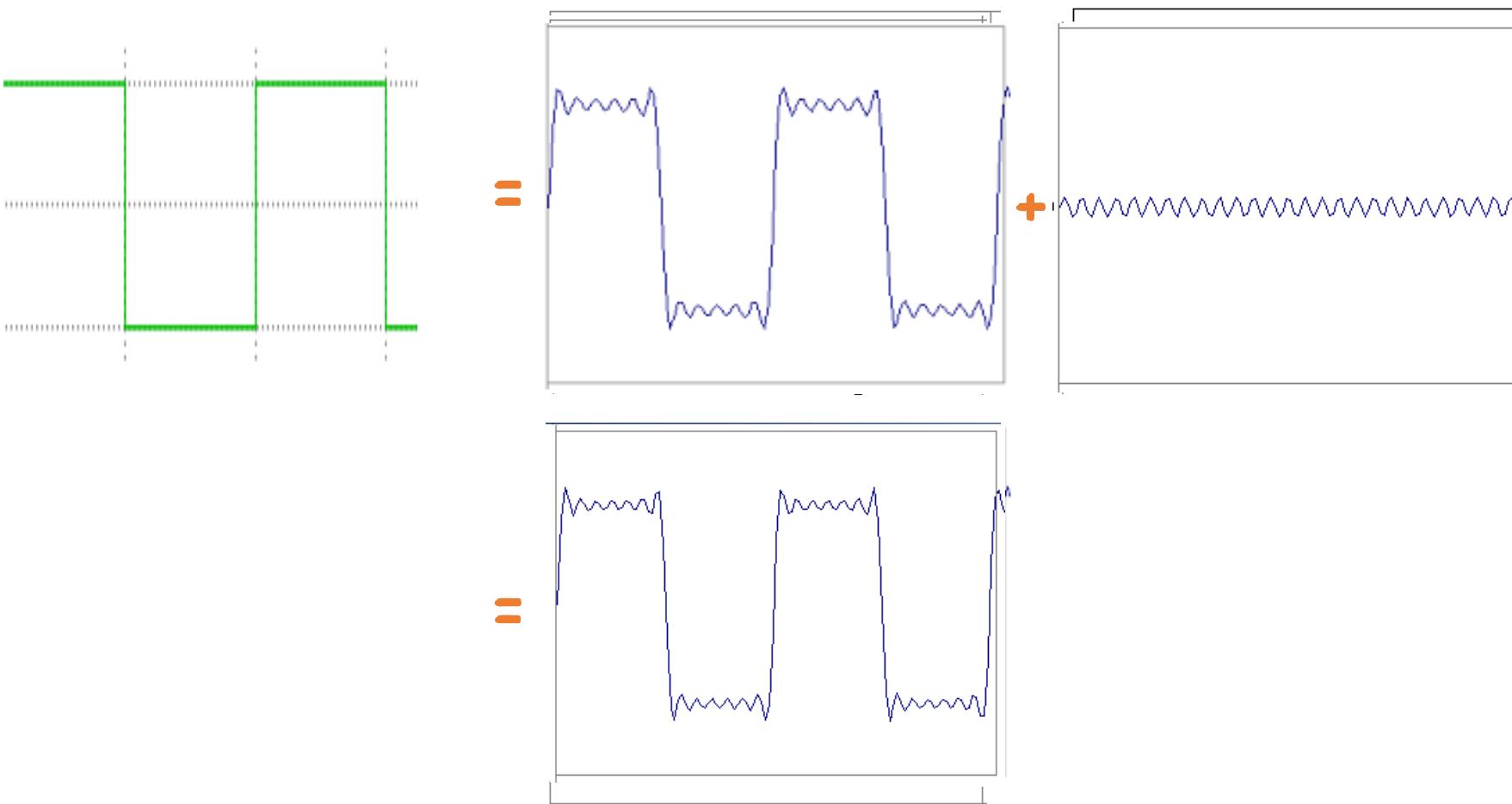
Frequency Spectra

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

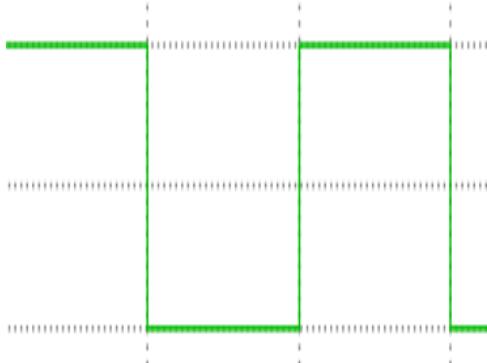


Slides: Efros

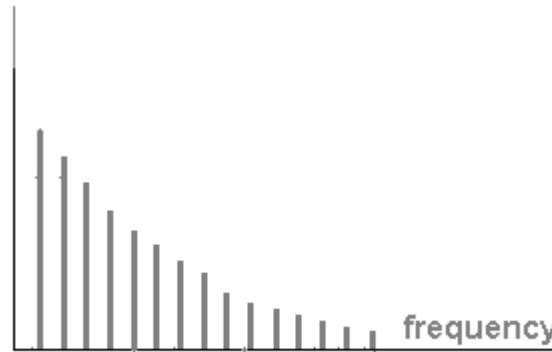
Frequency Spectra



Frequency Spectra

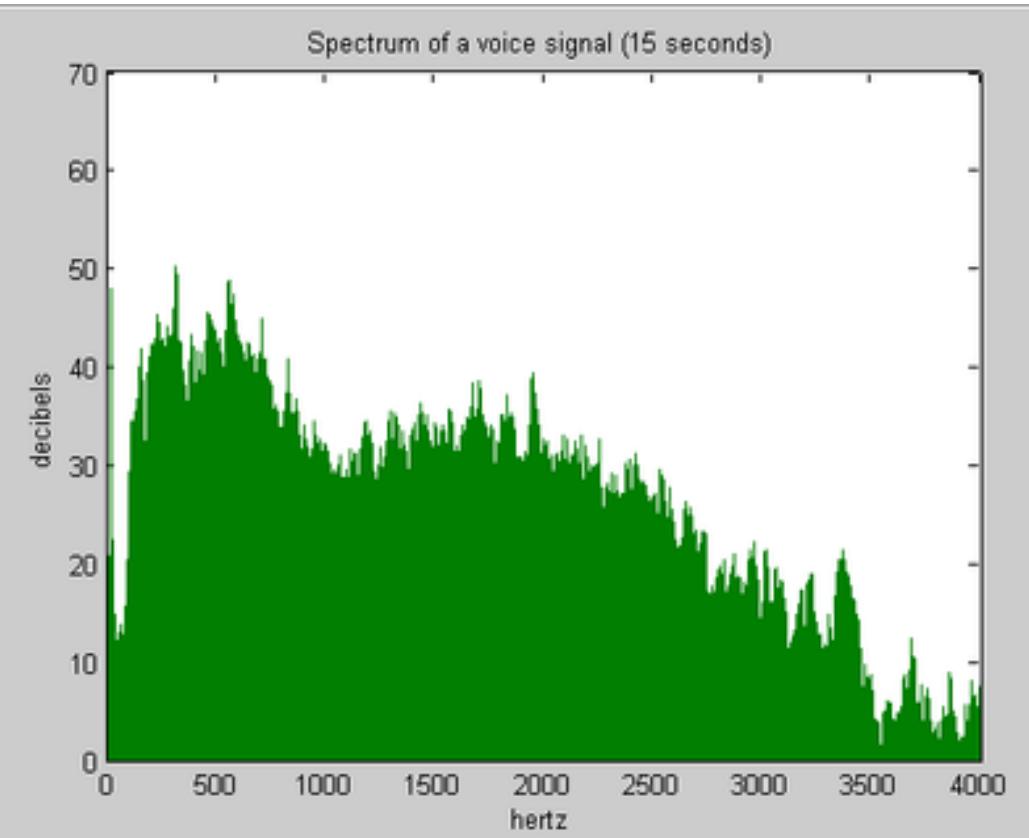
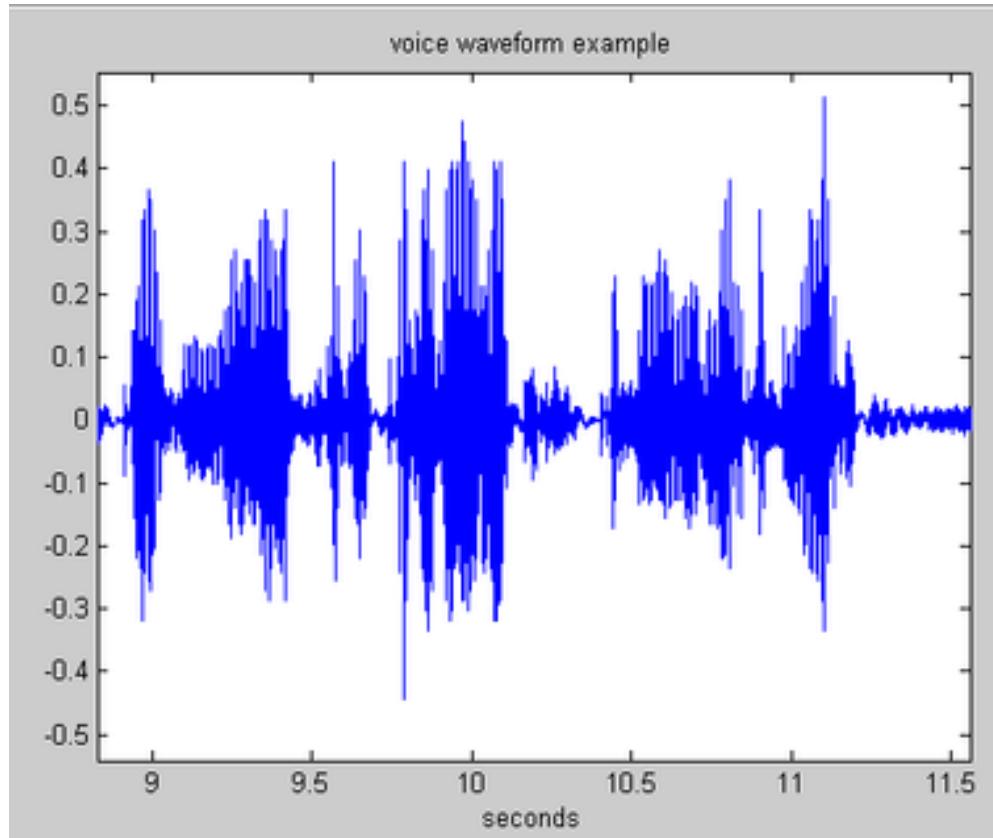


$$= A \sum_{k=1}^{\infty} \frac{1}{k} \sin(2\pi kt)$$



Example: Music

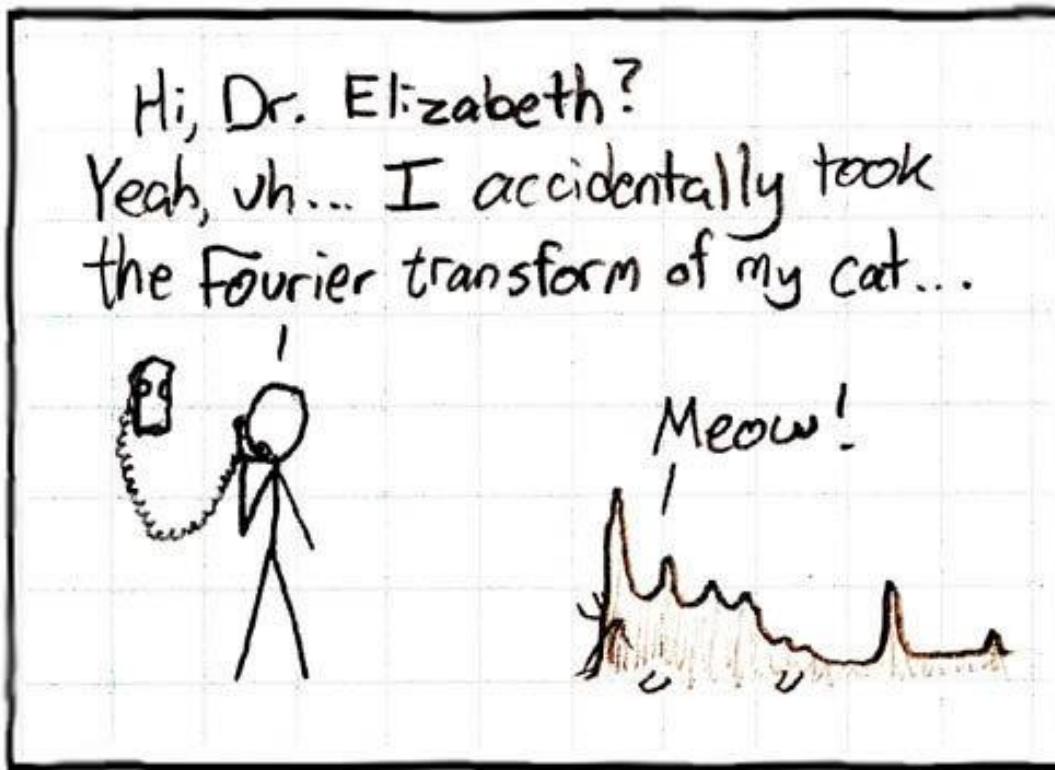
- We think of music in terms of frequencies at different magnitudes



Other Signals

- We can also think of all kinds of other signals the same way

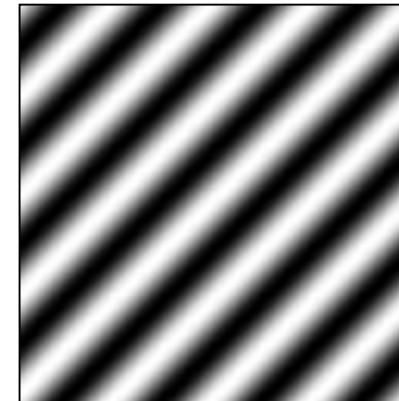
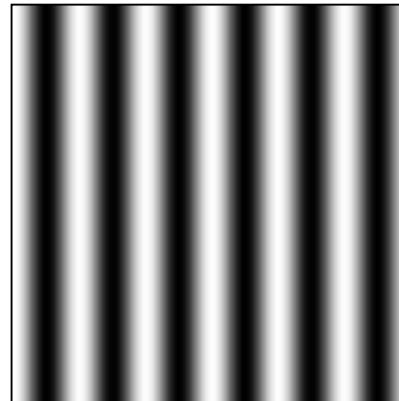
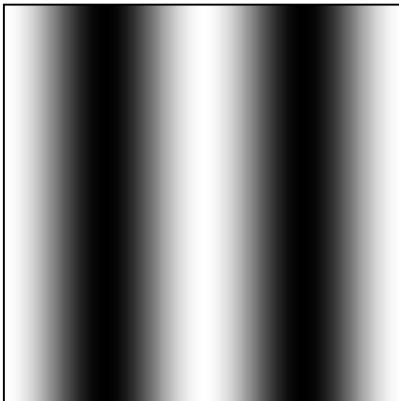
Cats(?)



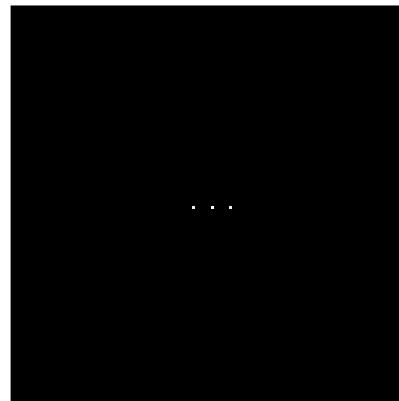
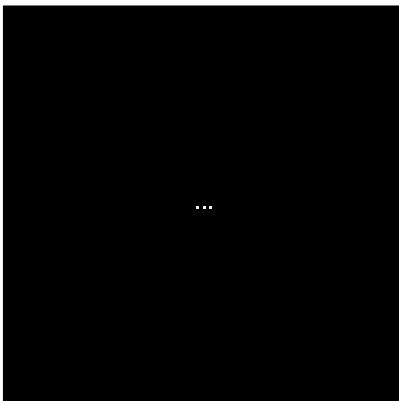
xkcd.com

Fourier Analysis in Images

Intensity
Image

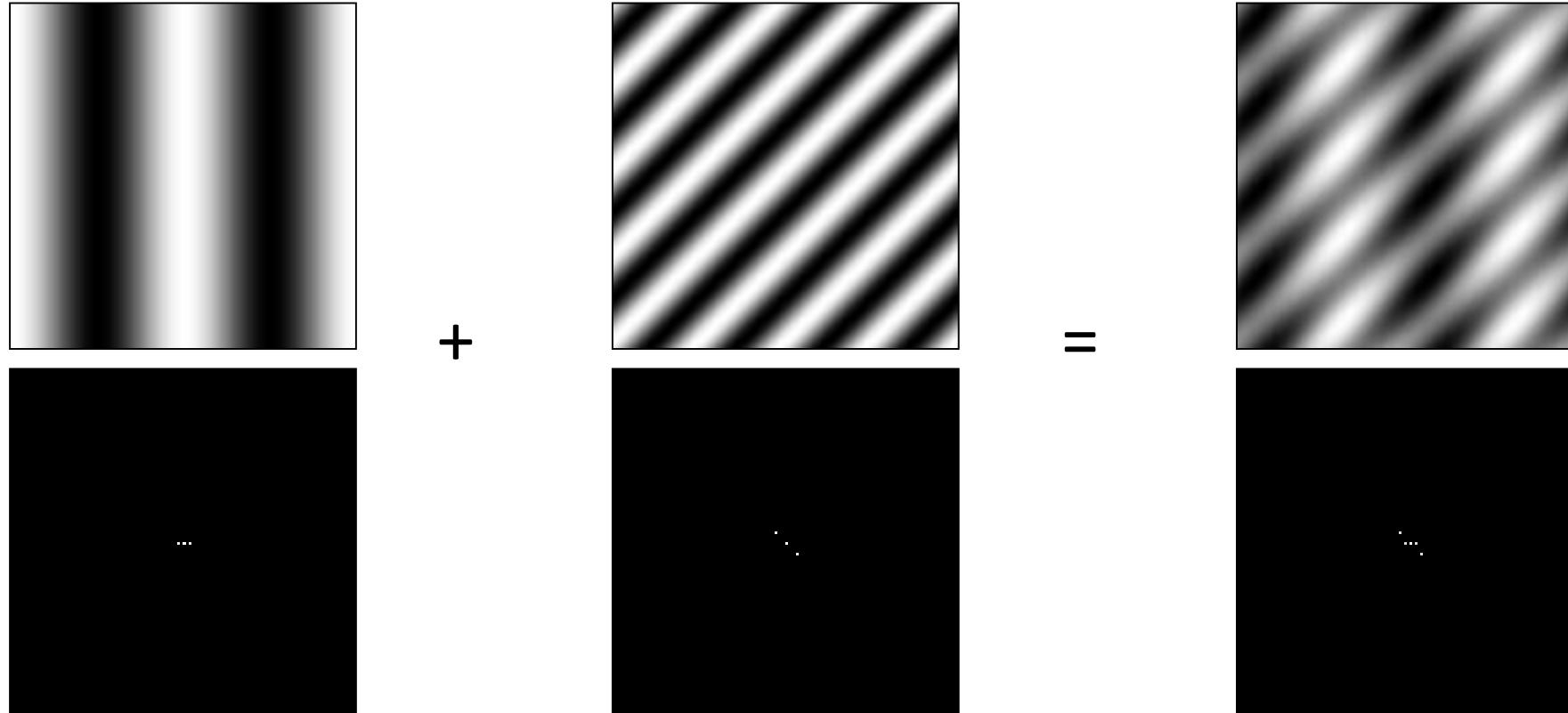


Fourier
Image



<http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering>

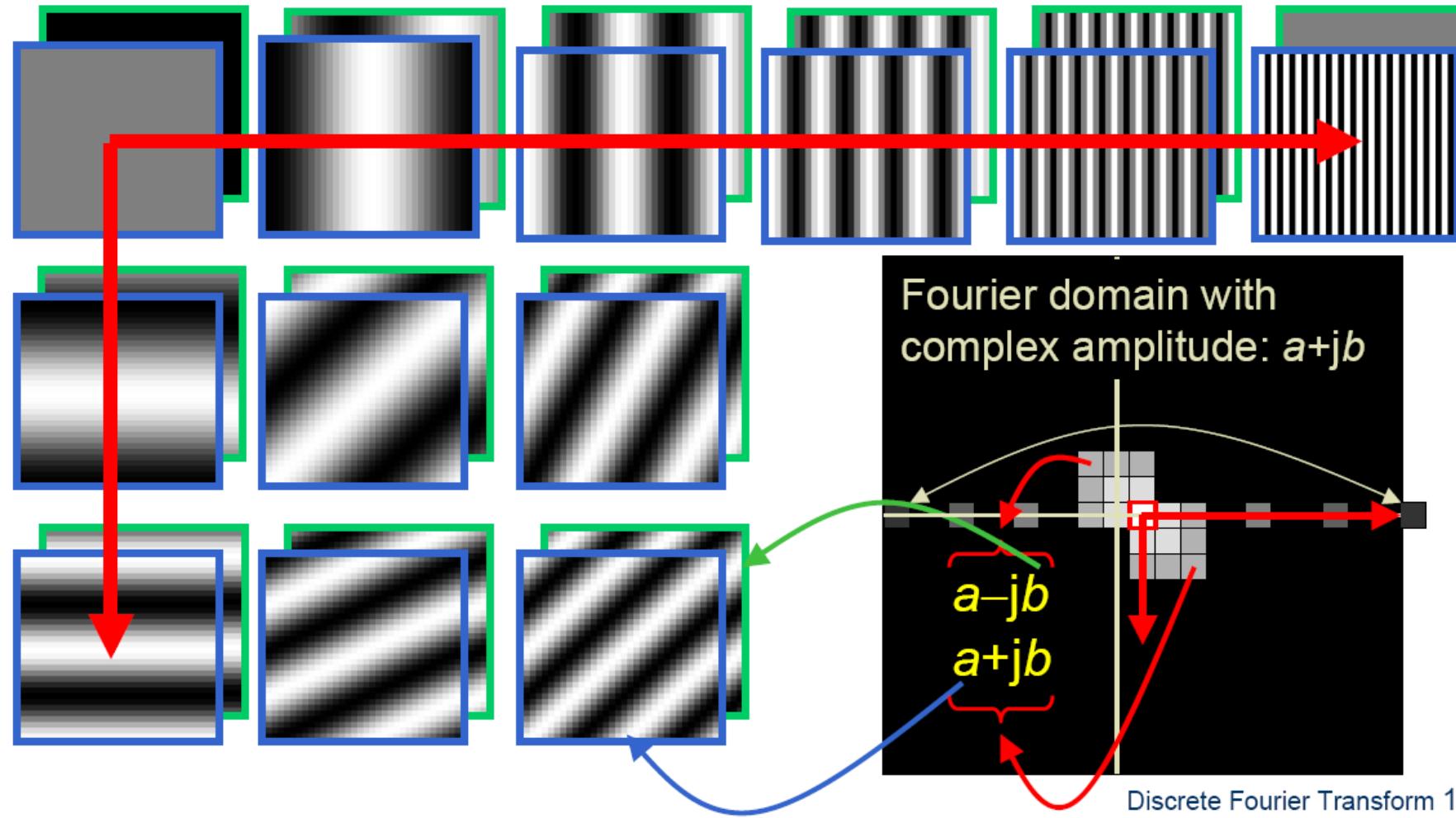
Signals can be composed



<http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering>
More: <http://www.cs.unm.edu/~brayer/vision/fourier.html>

Fourier Transform

Fourier analysis is a method by which any two-dimensional luminance image can be analyzed into the sum of a set of sinusoidal gratings that differ in spatial frequency, orientation, amplitude and phase.

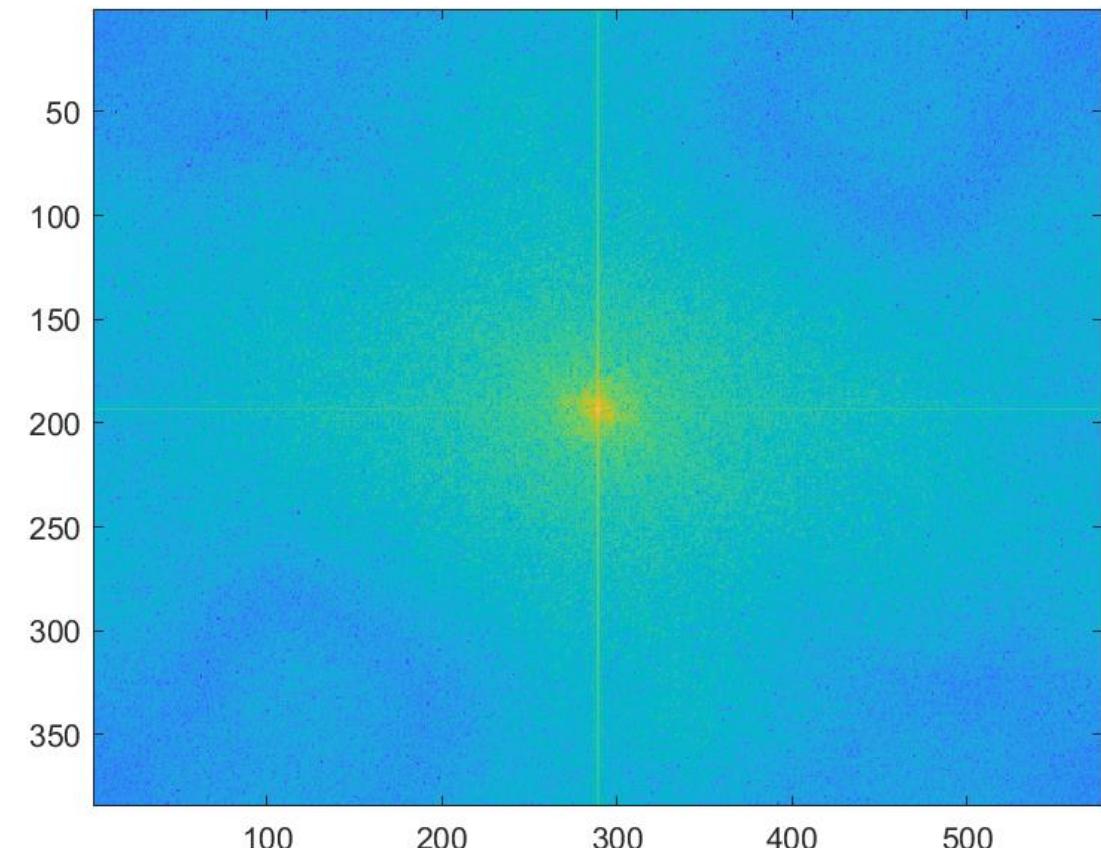


Teases away fast vs.
slow changes in the
image.

Image as a sum
of basis images

Example

- `im = imread('0901.png');`
- `im = rgb2gray(im);`
- `imagesc(log(abs(fftshift(fft2(im)))));`



2D Fourier Transform

Fourier Transform:

$$F(u, v) = \iint_{-\infty}^{\infty} f(x, y) e^{-i2\pi(ux+vy)} dx dy$$

u and v are frequencies along x and y , respectively

Inverse Fourier Transform:

$$f(x, y) = \iint_{-\infty}^{\infty} F(u, v) e^{i2\pi(xu+yv)} du dv$$

Discrete Fourier Transform (DFT):

$$F[p, q] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f[m, n] e^{-i2\pi pm/M} e^{-i2\pi qn/N}$$

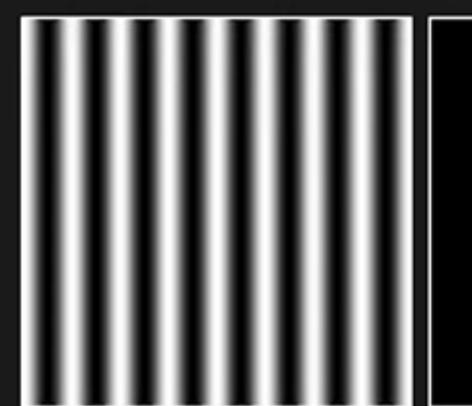
p and q are frequencies along m and n , respectively

Inverse Discrete Fourier Transform (IDFT):

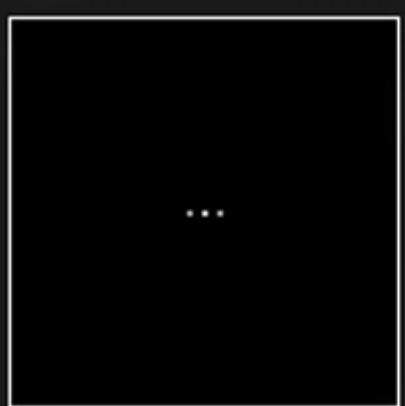
$$f[m, n] = \frac{1}{MN} \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} F[p, q] e^{i2\pi pm/M} e^{i2\pi qn/N}$$



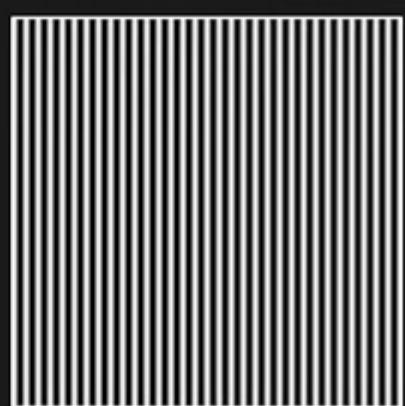
2D Fourier Transform : Example 1



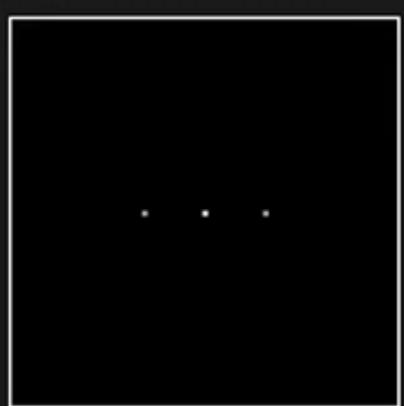
$f(m, n)$



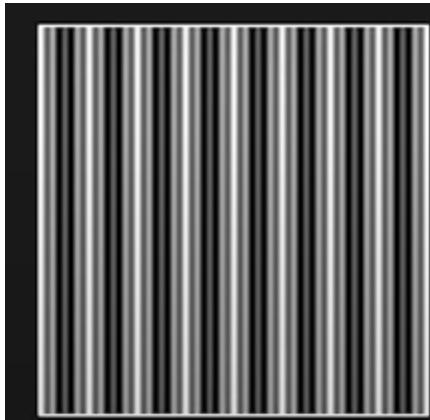
$\log(|F(p, q)|)$



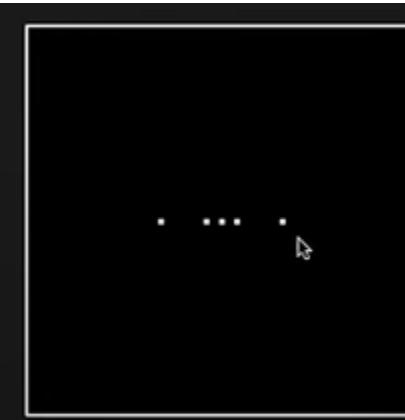
$g(m, n)$



$\log(|G(p, q)|)$

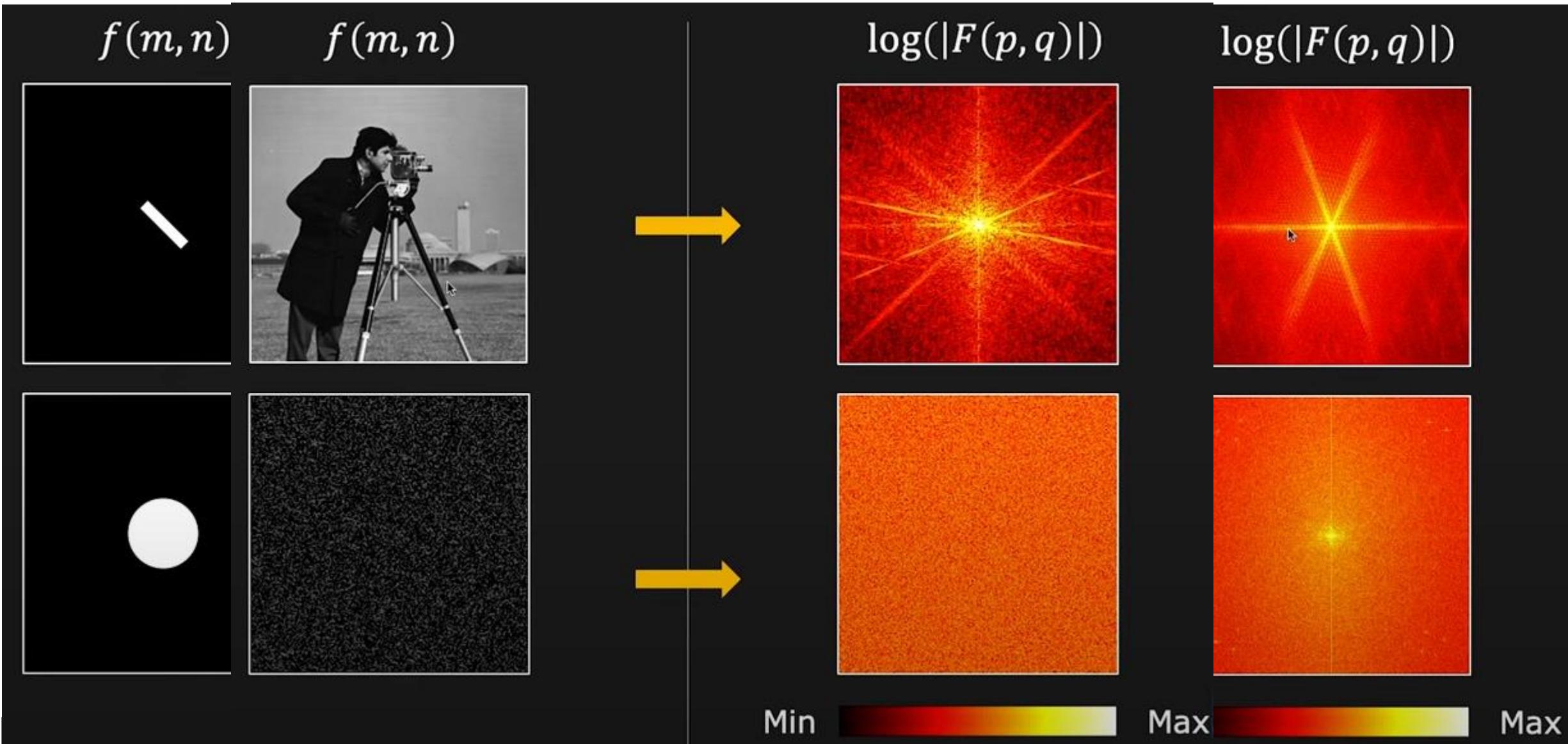


$f(m, n) + g(m, n)$



$\log(|F(p, q) + G(p, q)|)$

2D Fourier Transform : Example 1



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

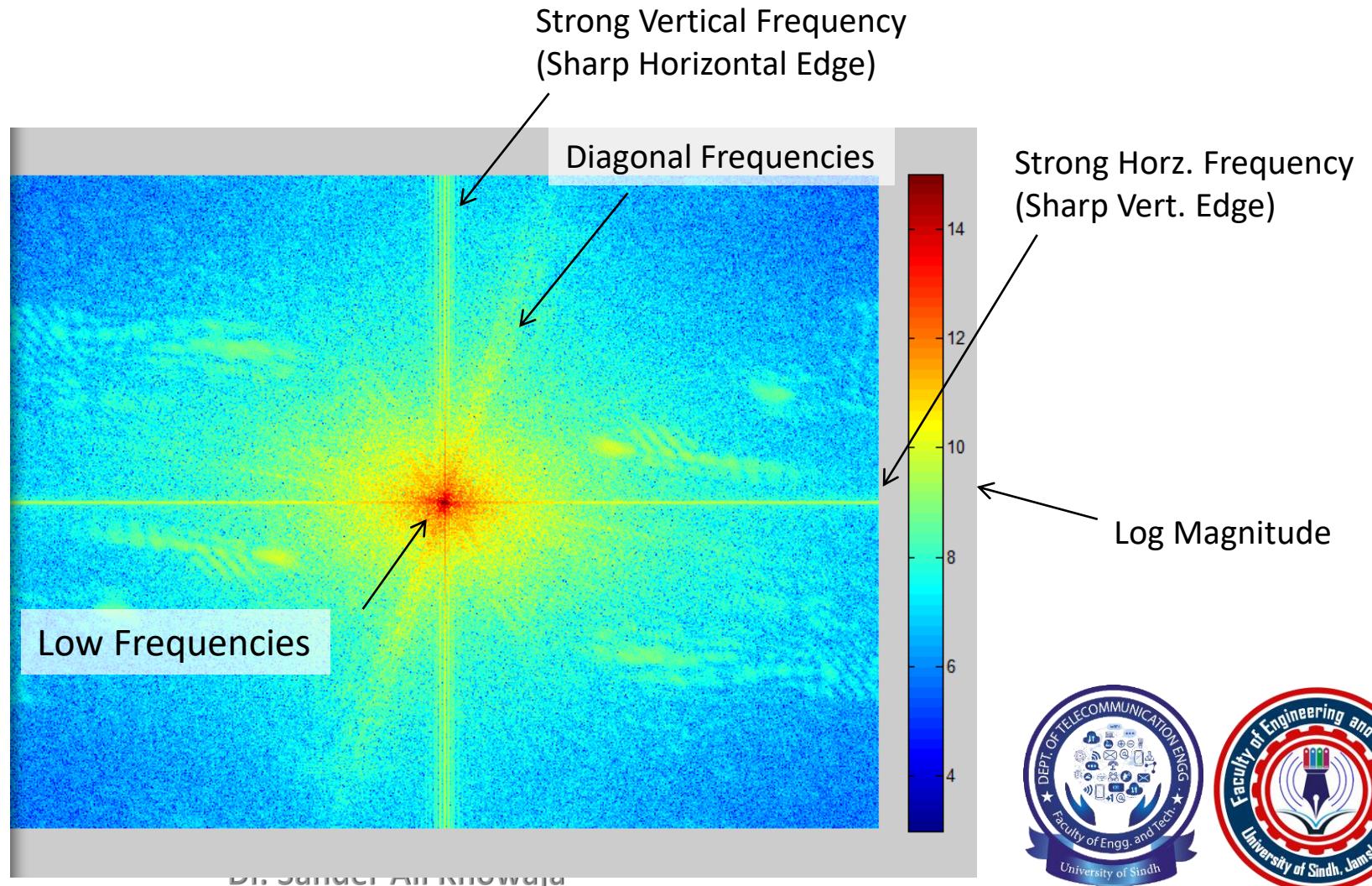
Amplitude: $A = \pm \sqrt{R(\omega)^2 + I(\omega)^2}$

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

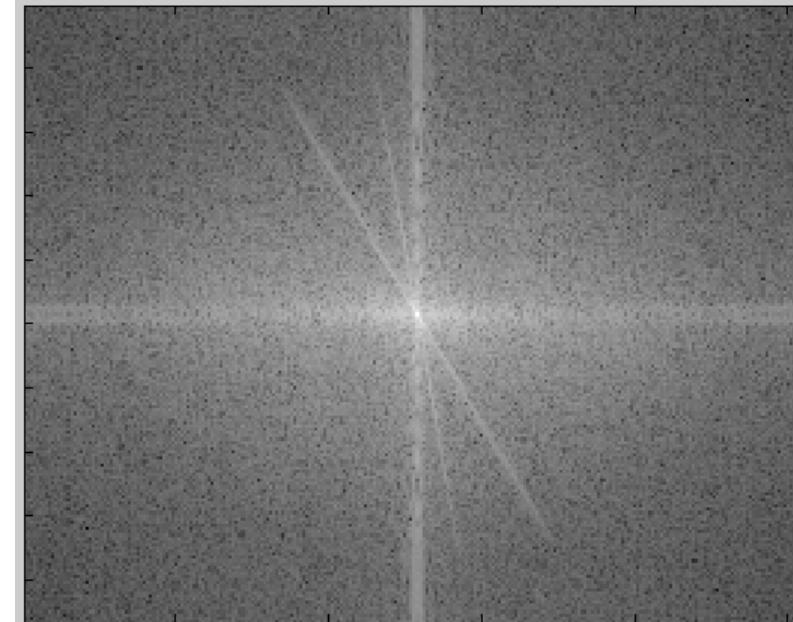
Euler's formula: $e^{inx} = \cos(nx) + i \sin(nx)$



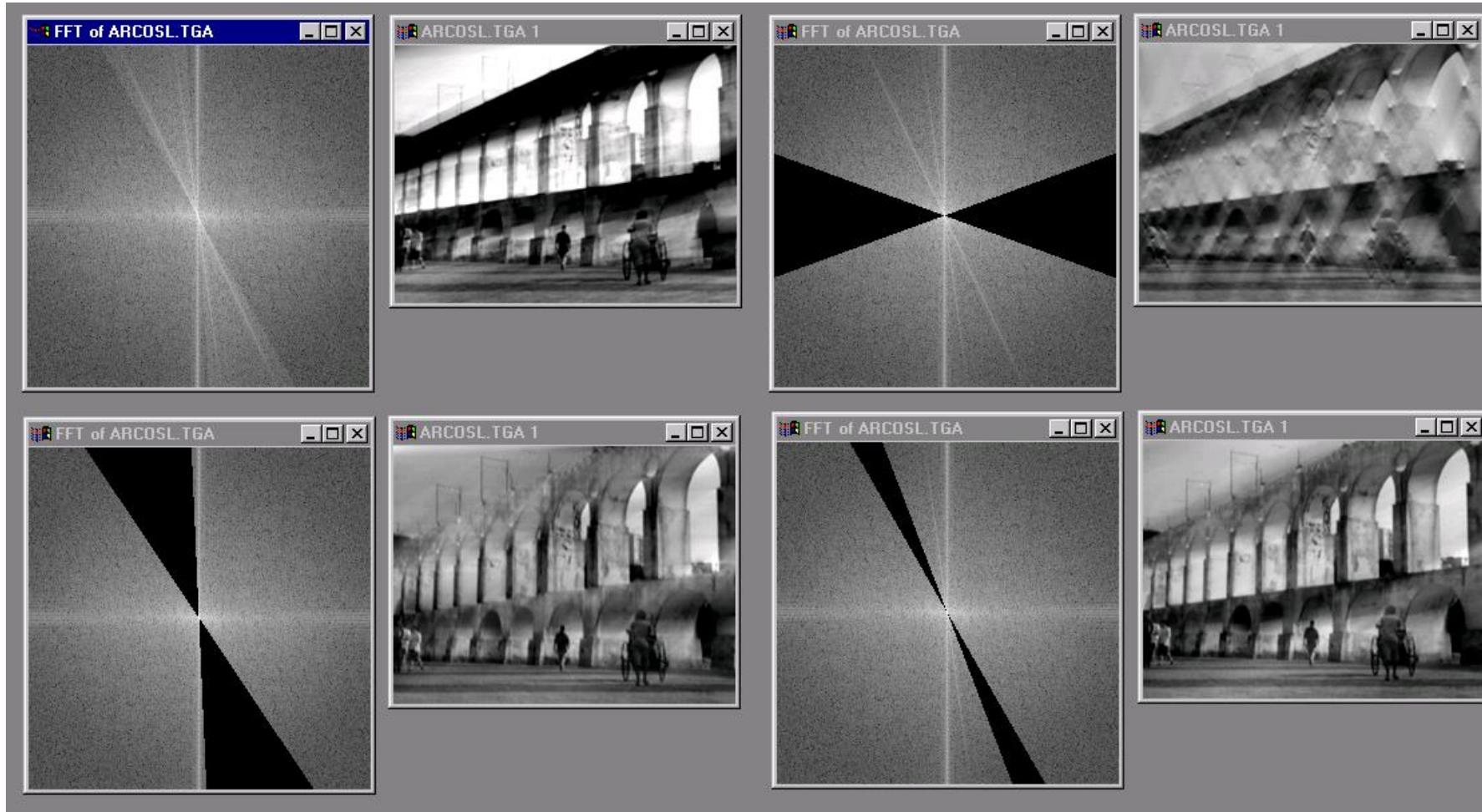
Interpreting Magnitude Images



Man-made Scene



Chan Change Spectrum, then reconstruct



Low Pass Filtering

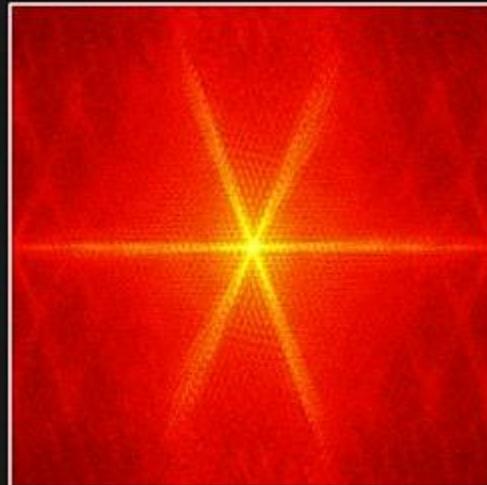
$f(m, n)$



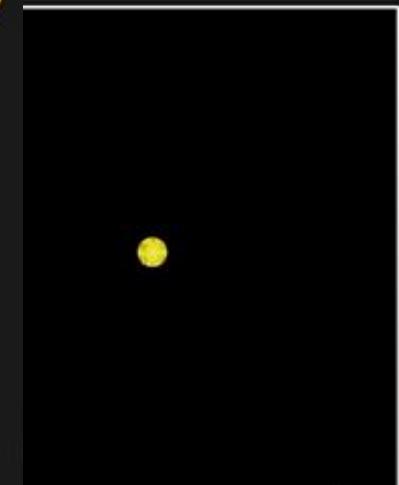
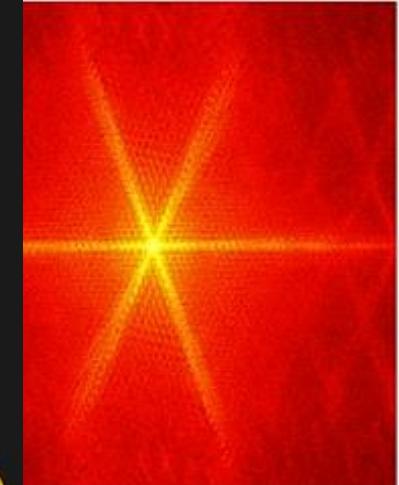
$f(m, n)$



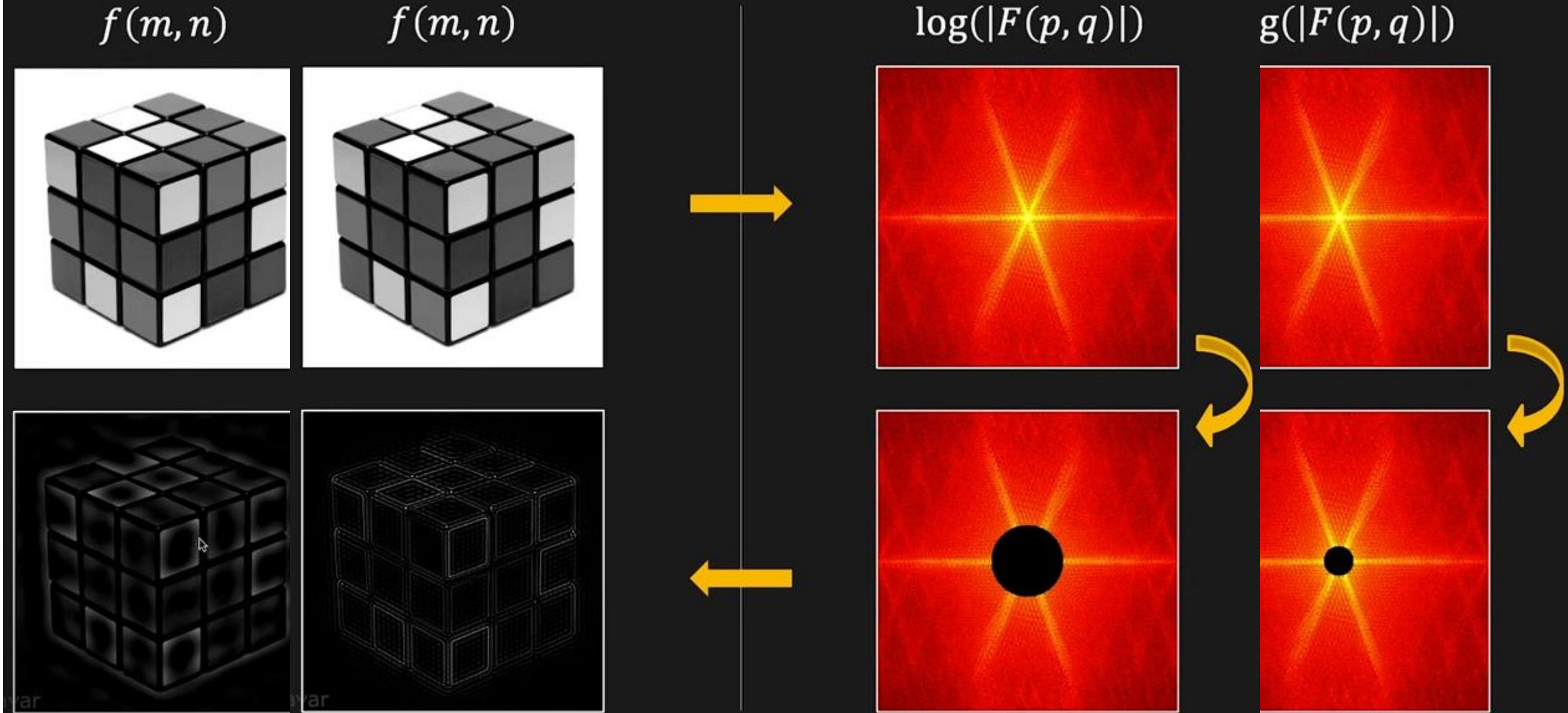
$\log(|F(p, q)|)$



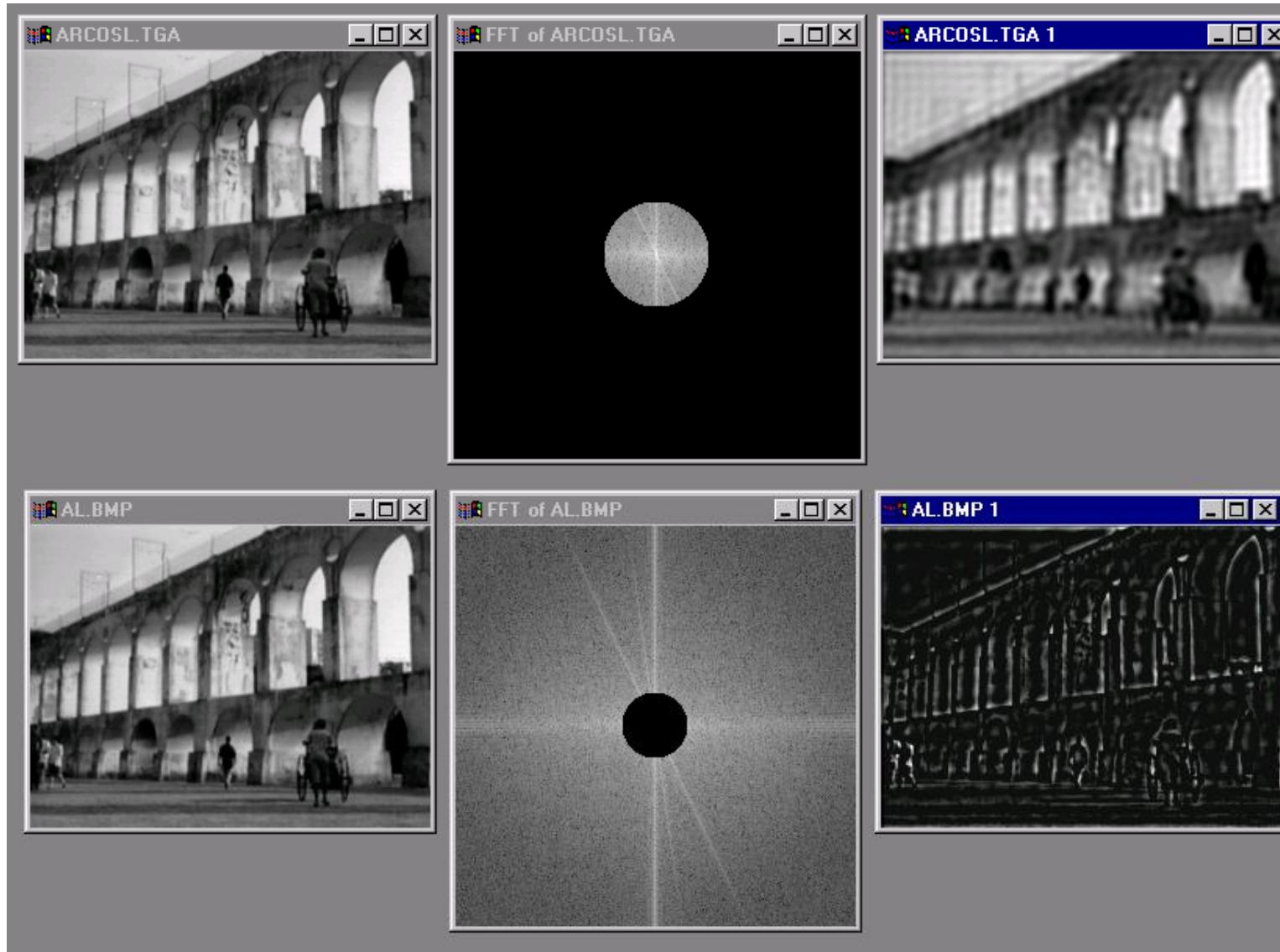
$g(|F(p, q)|)$



High Pass Filtering



Low and High Pass Filters



Dr. Sander Ali Khowaja



The Convolution Theorem

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

$$F[g * h] = F[g]F[h]$$

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

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

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



Properties of Fourier Transform

- 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

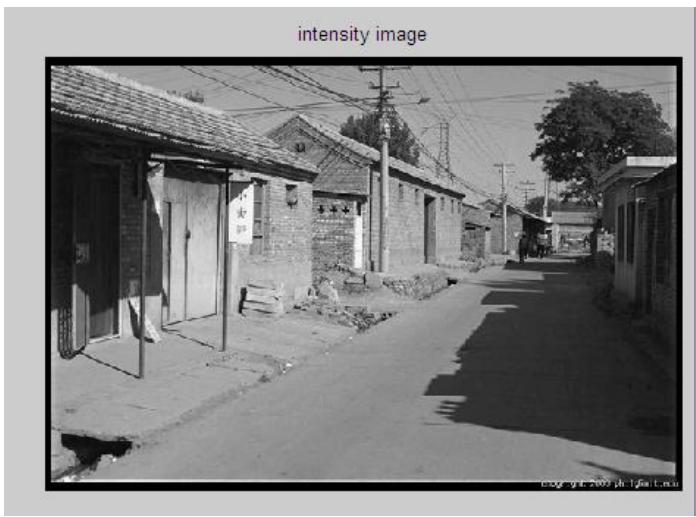
See Szeliski Book (3.4)

Dr. Sander Ali Khowaja



Filtering in Spatial Domain

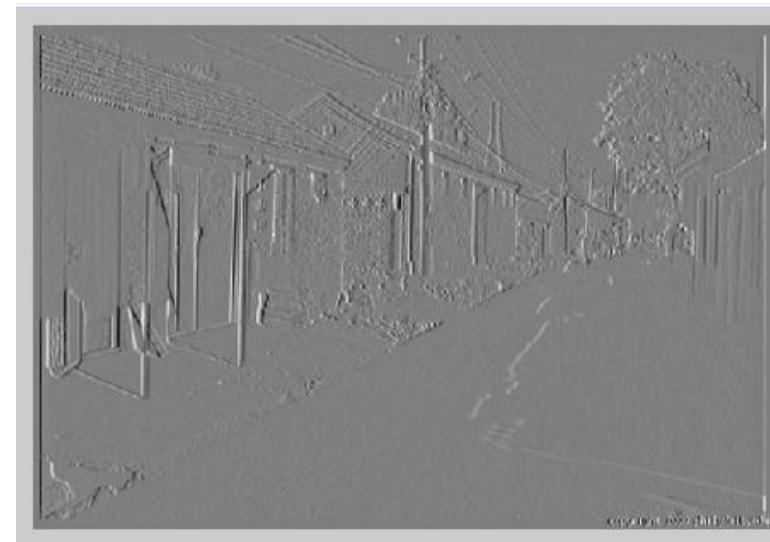
1	0	-1
2	0	-2
1	0	-1



$$\ast =$$

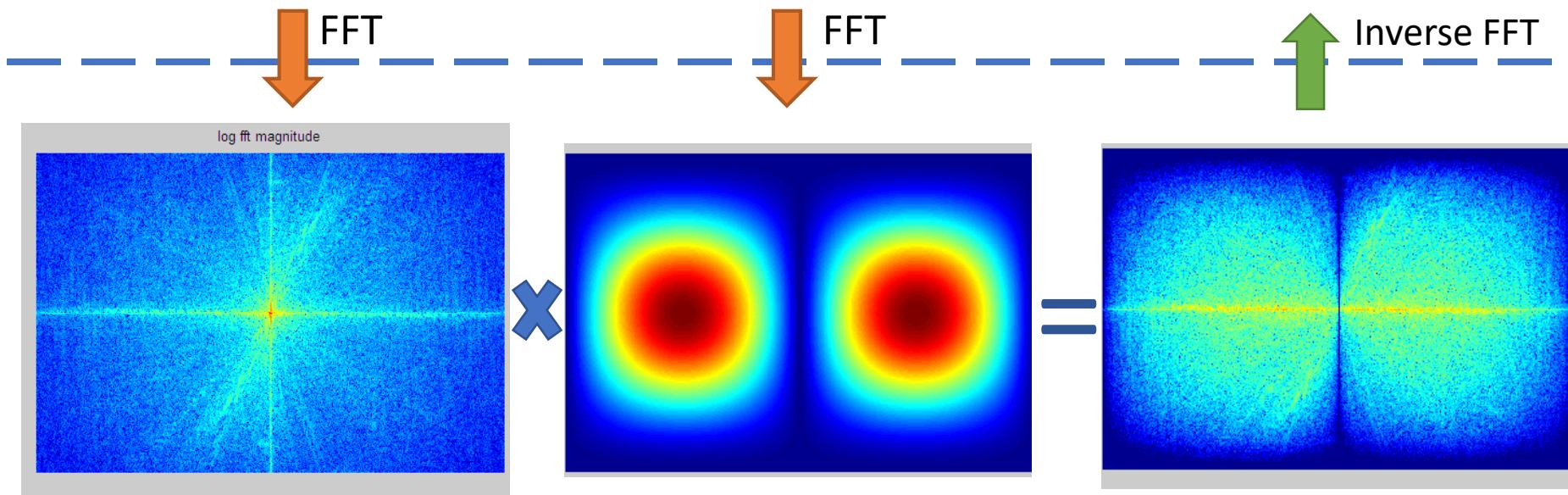
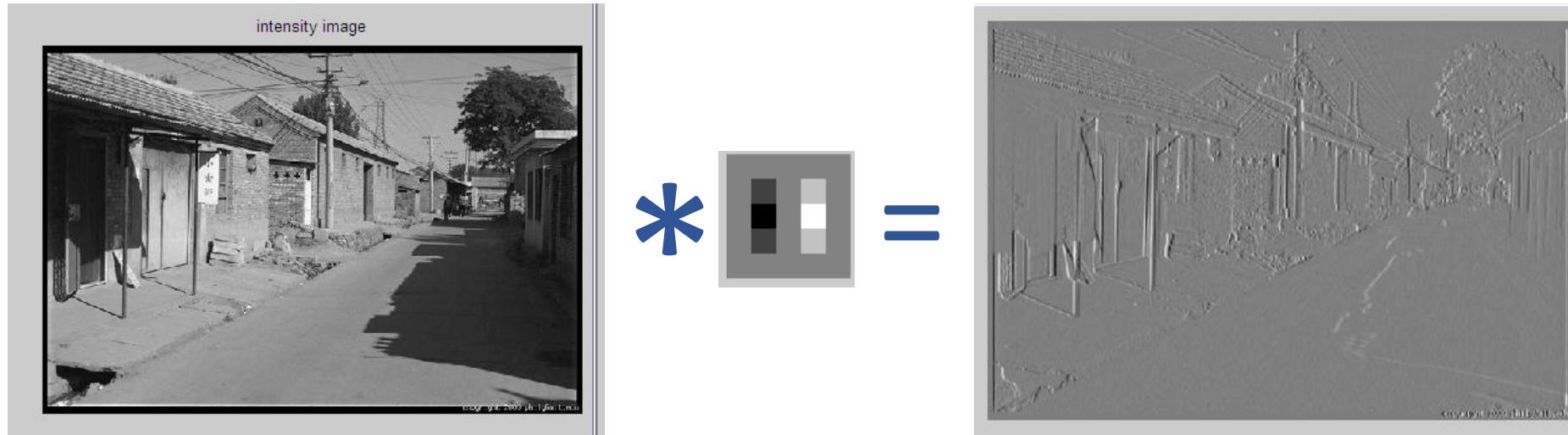
A 3x3 kernel for edge detection, represented as a matrix:

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$



Filtering in Fourier Domain

Spatial domain

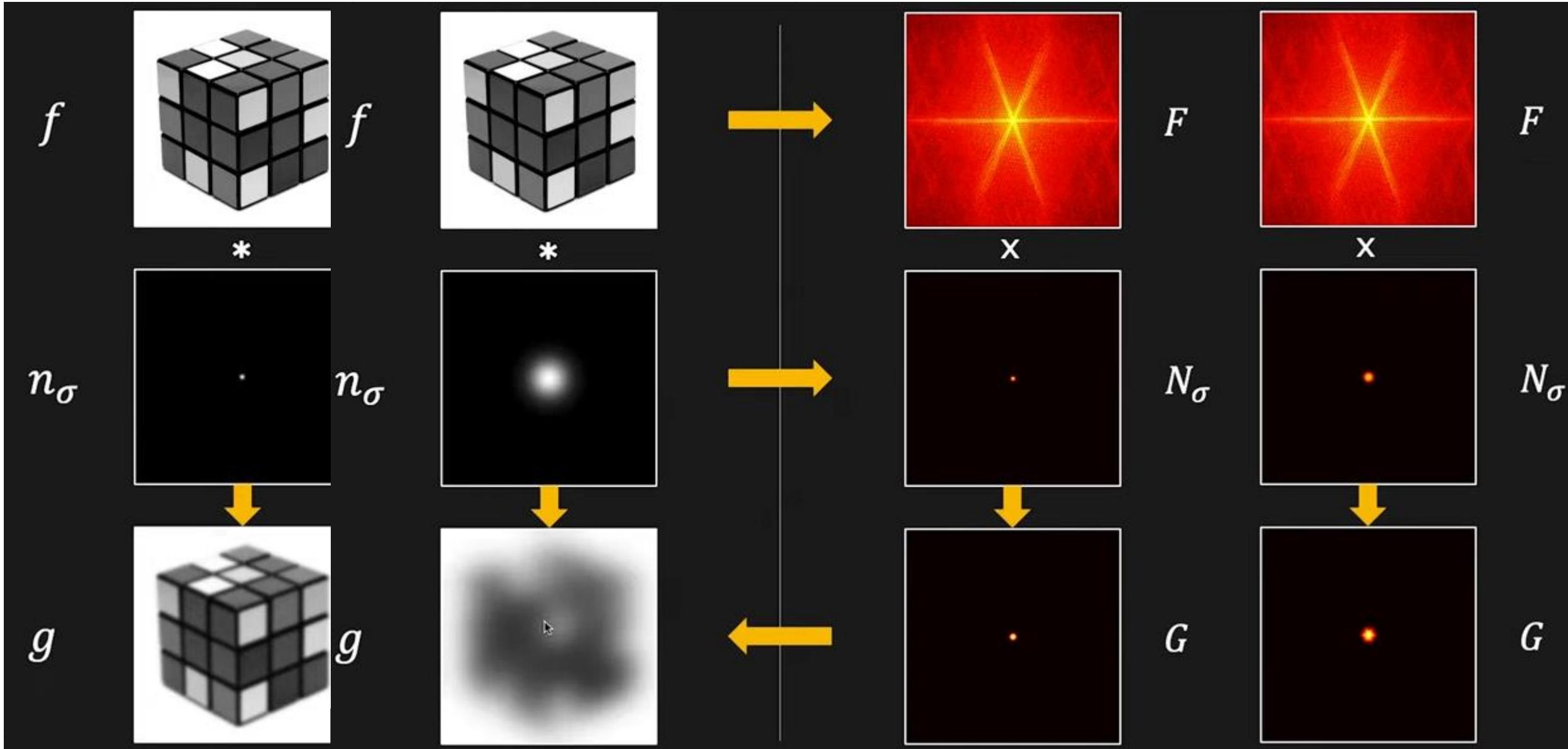


Frequency domain

Dr. Sander Ali Khowaja

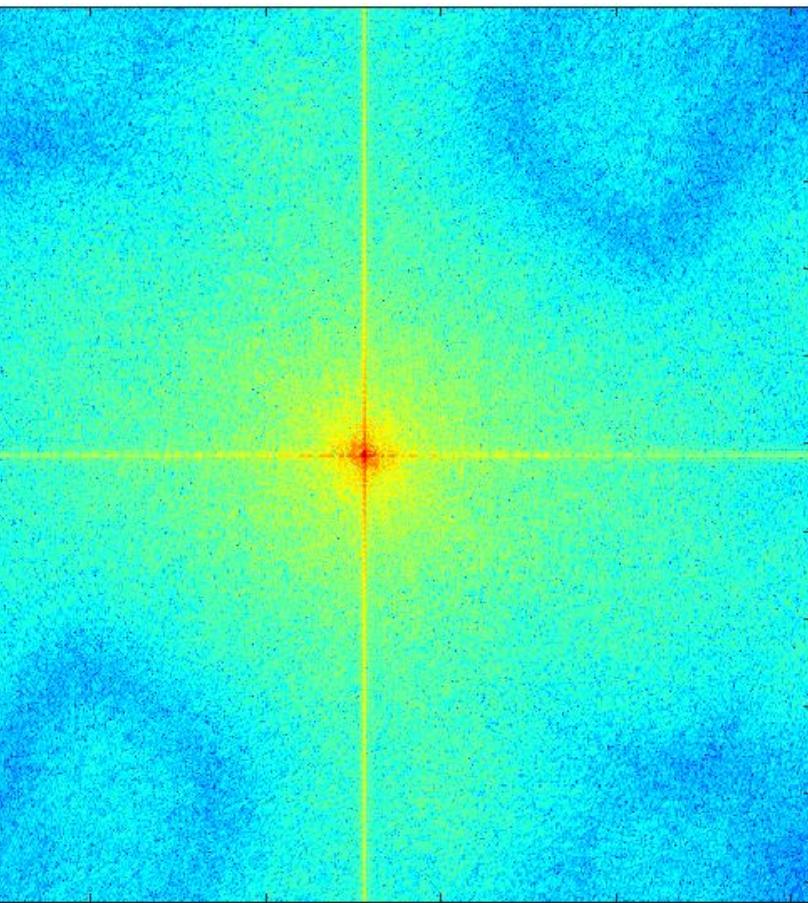


Gaussian Smoothing in Frequency Domain



FFT based Gaussian Filtering Example

```
1 im = rgb2gray(imread('0901.png'));
2 im = im2double(im);
3 [imh, imw] = size(im);
4 fftsize = 1024;
5 hs = 50;
6 fil = fspecial('gaussian',hs*2+1,10);
7 im_fft = fft2(im,fftsize,fftsize);
8 fil_fft = fft2(fil,fftsize,fftsize);
9 im_fil_fft = im_fft.*fil_fft;
10 im_fil = ifft2(im_fil_fft);
11 im_fil = im_fil(1+hs:size(im,1)+hs, 1+hs:size(im,2)+hs);
12 imagesc(log(abs(fftshift(im_fft))), axis image, colormap jet)
```

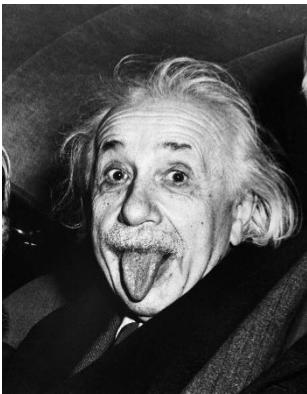


Which has more information, the phase or the magnitude?

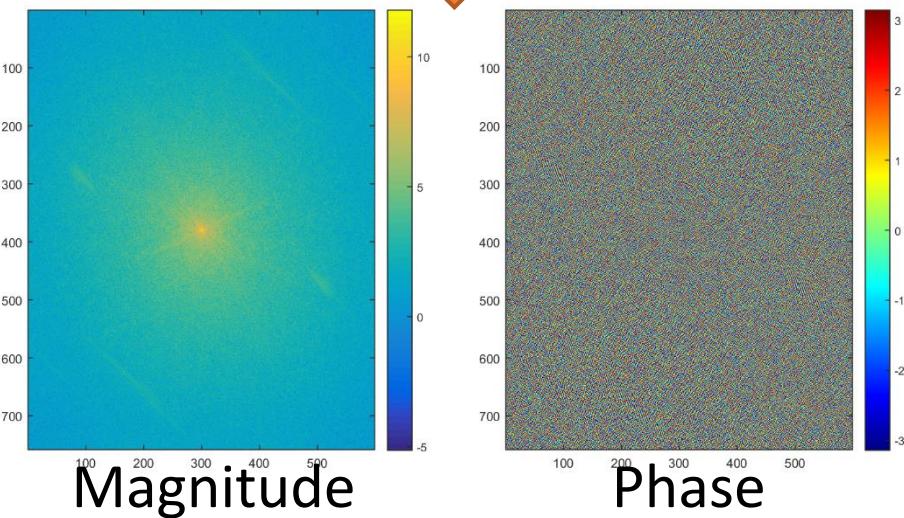
What happens if you take the phase from one image and combine it with the magnitude from another image?



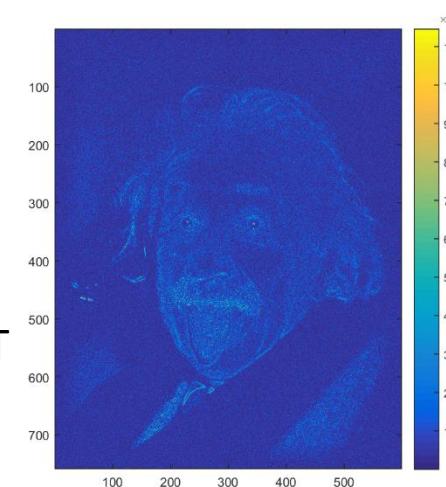
Phase vs Magnitude



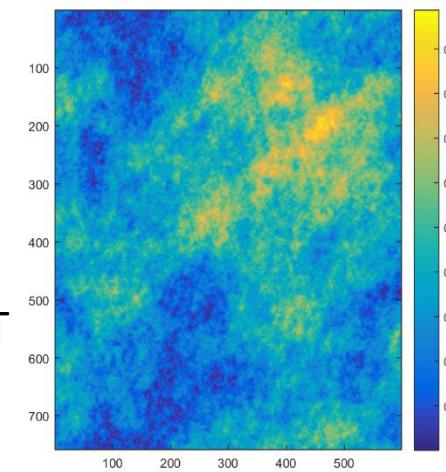
Intensity image
FFT



Use random magnitude
Inverse FFT



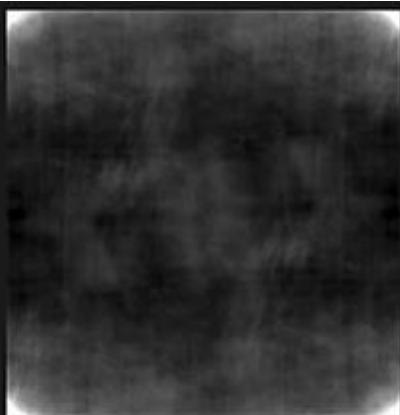
Use random phase
Inverse FFT



Importance of Phase



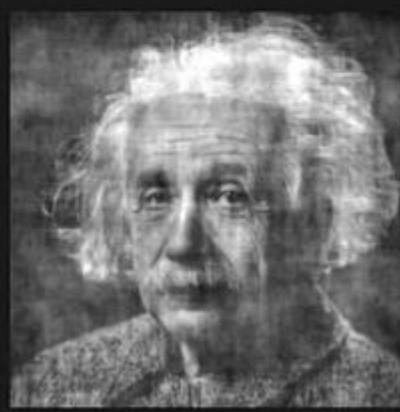
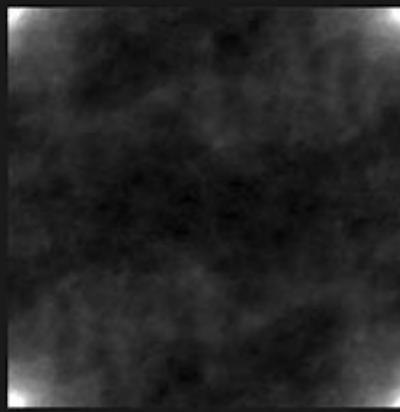
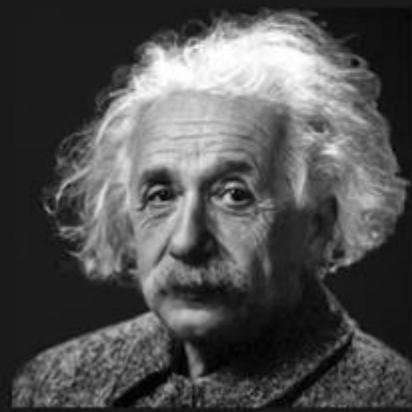
Original Image



Magnitude Preserved,
Phase Set to Zero

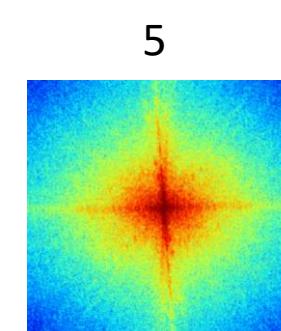
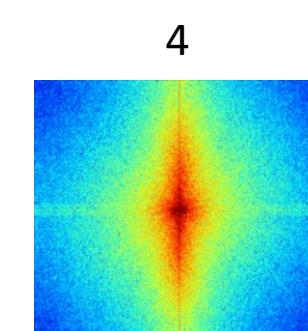
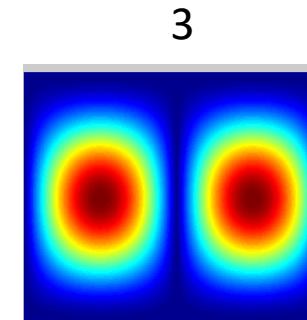
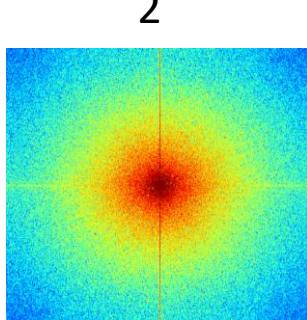
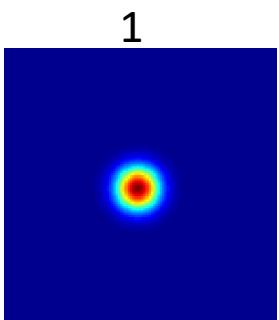


Phase Preserved,
Magnitude Set to Average
of Natural Images



Question

Match the spatial domain image to the Fourier magnitude image



B

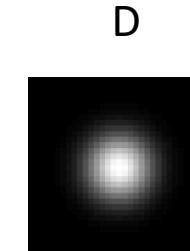
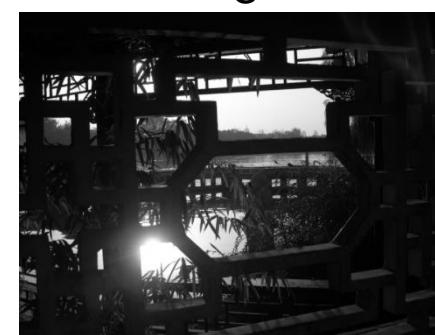
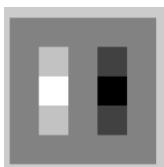


Image Half-sizing

This image is too big to fit on the screen. How can we reduce it?

How to generate a half-sized version?

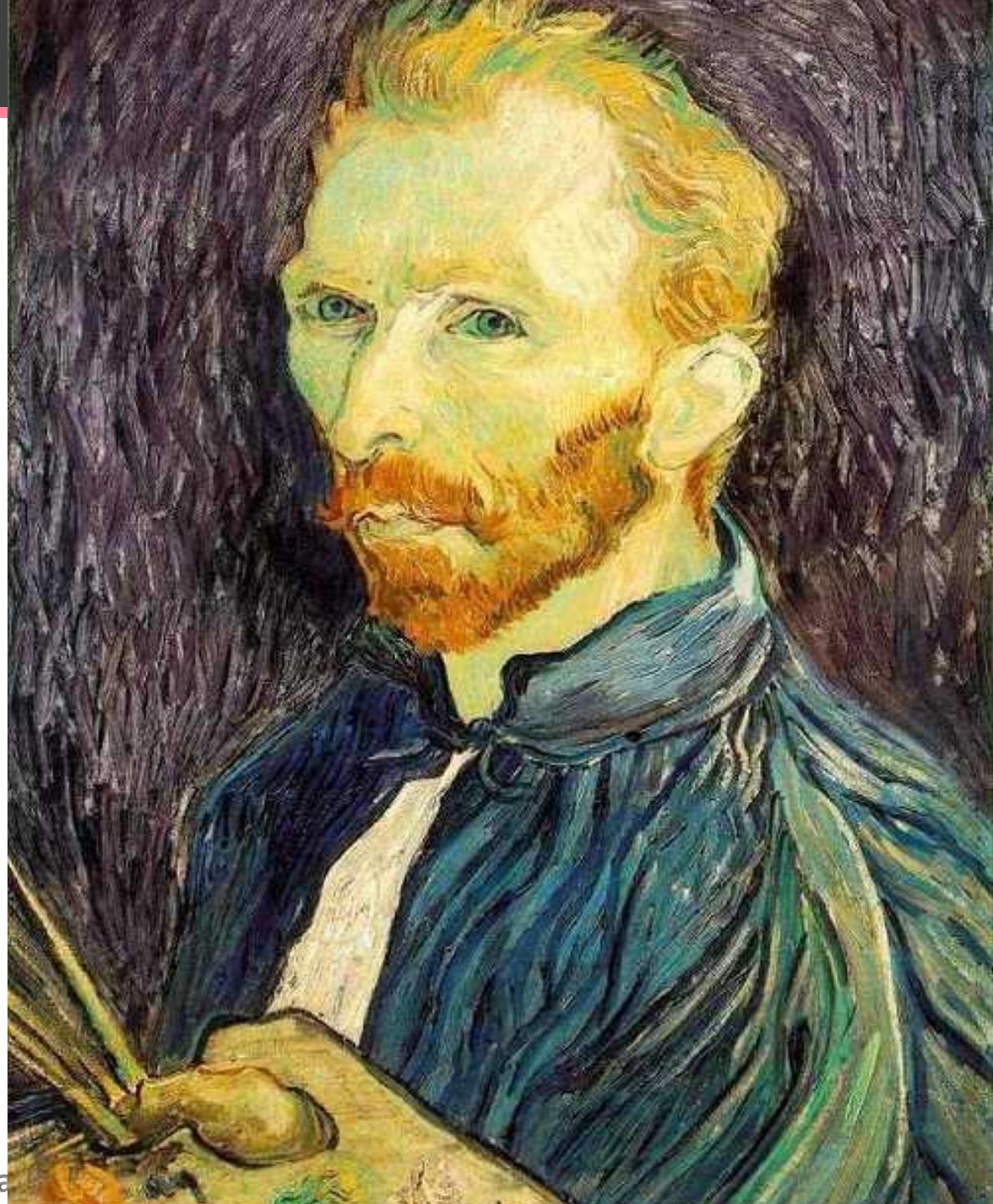
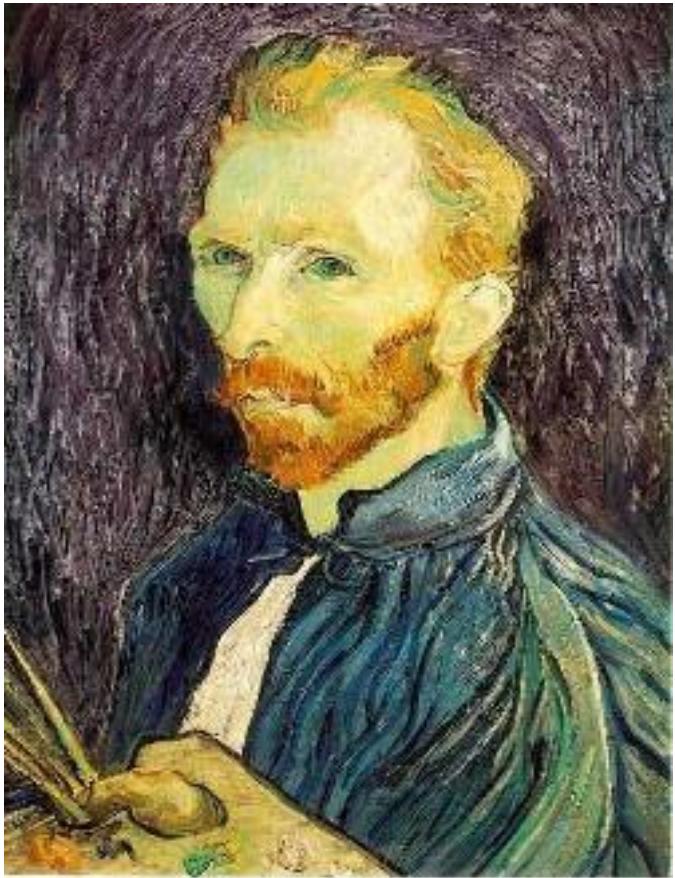


Image Sub-Sampling



1/4

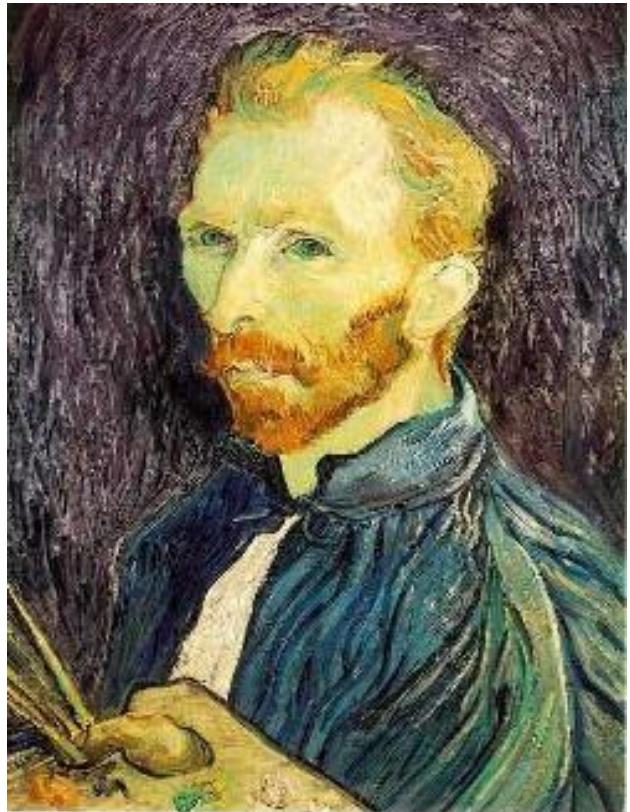


1/8

Throw away every other row and column to create a $1/2$ size image
- called *image sub-sampling*

Slide by Steve Seitz

Image Sub-Sampling



1/2



1/4 (2x zoom)

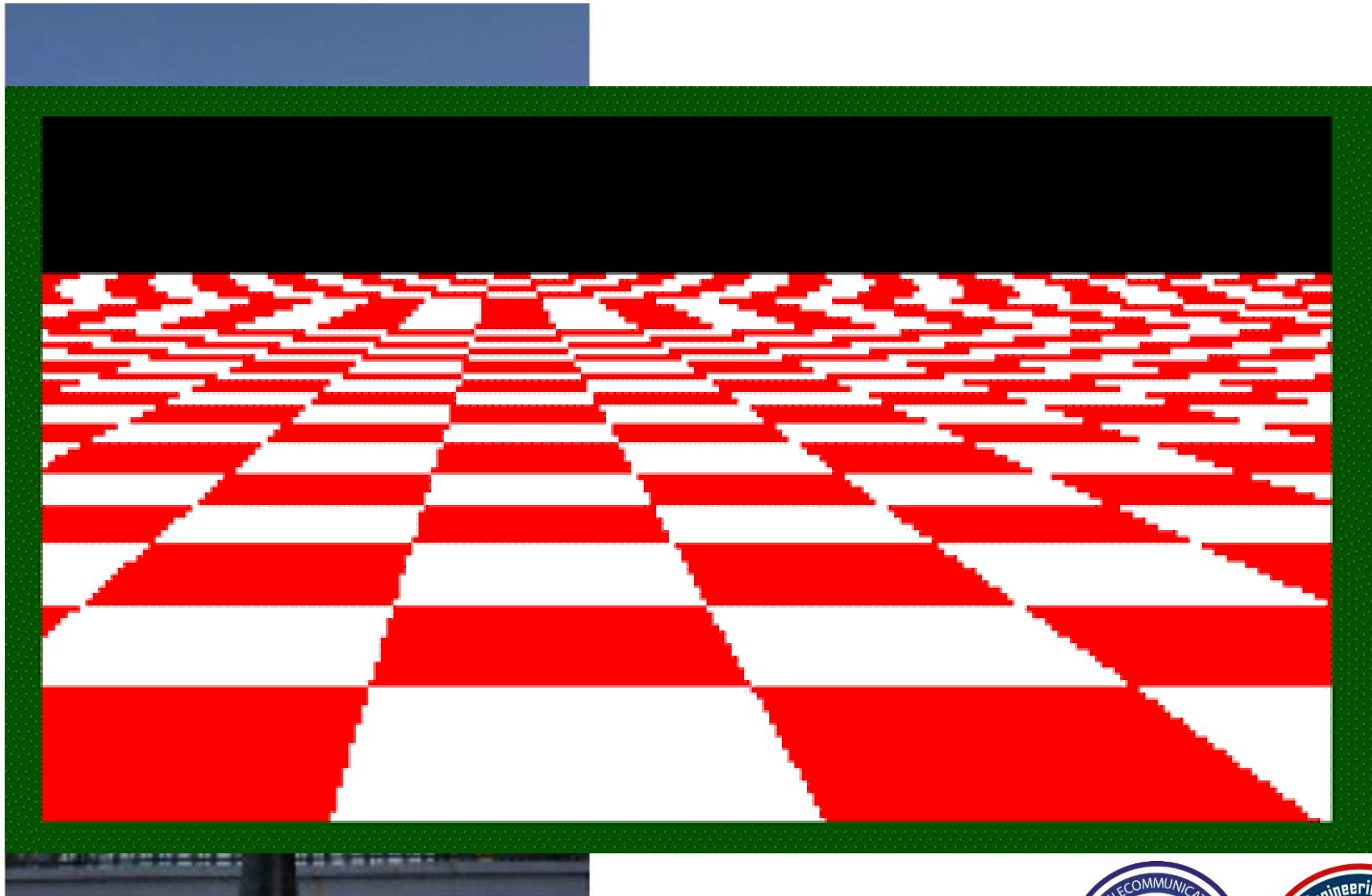


1/8 (4x zoom)

Why does this look so cruddy?
Aliasing! What do we do?

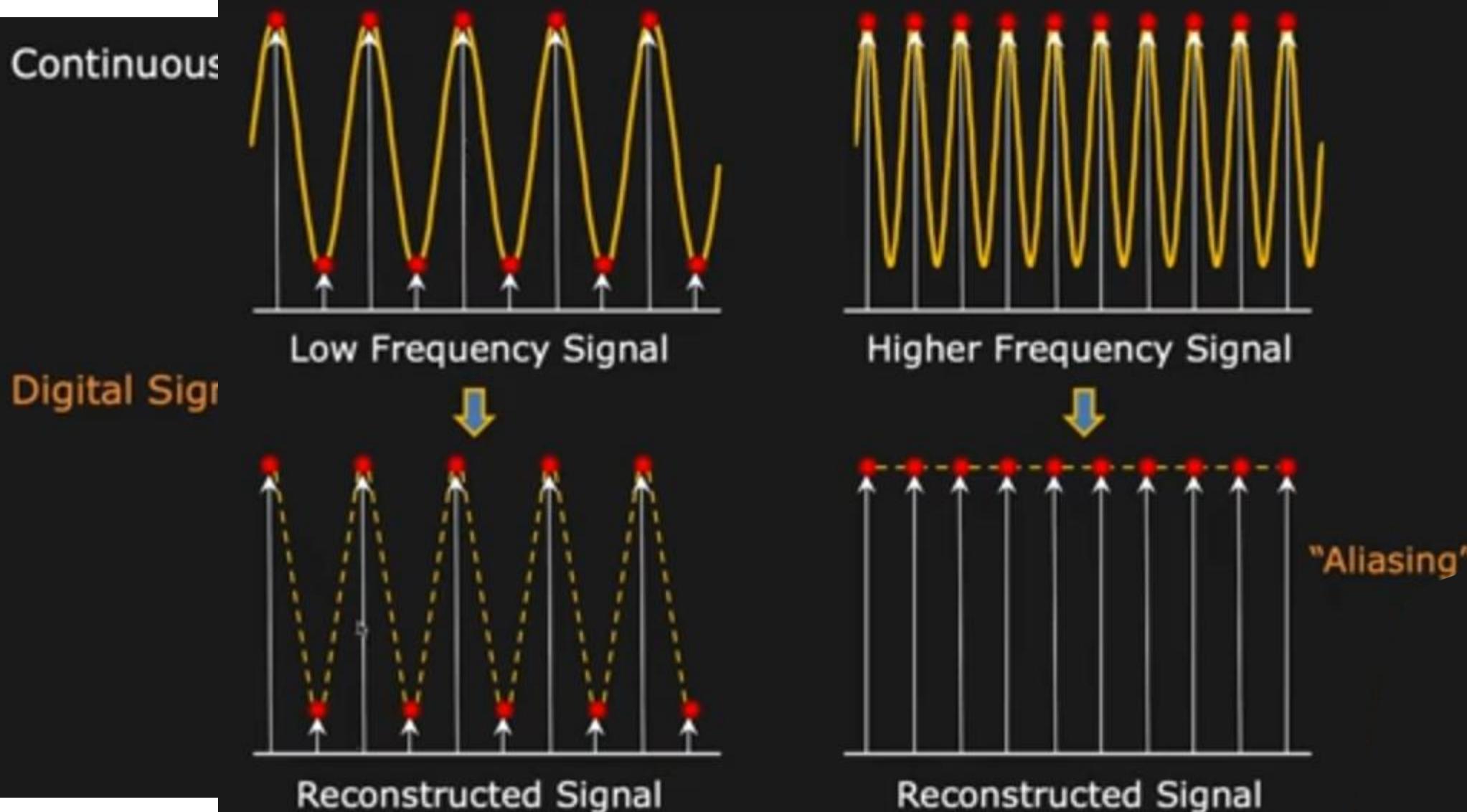
Slide by Steve Seitz

Image Sub-Sampling



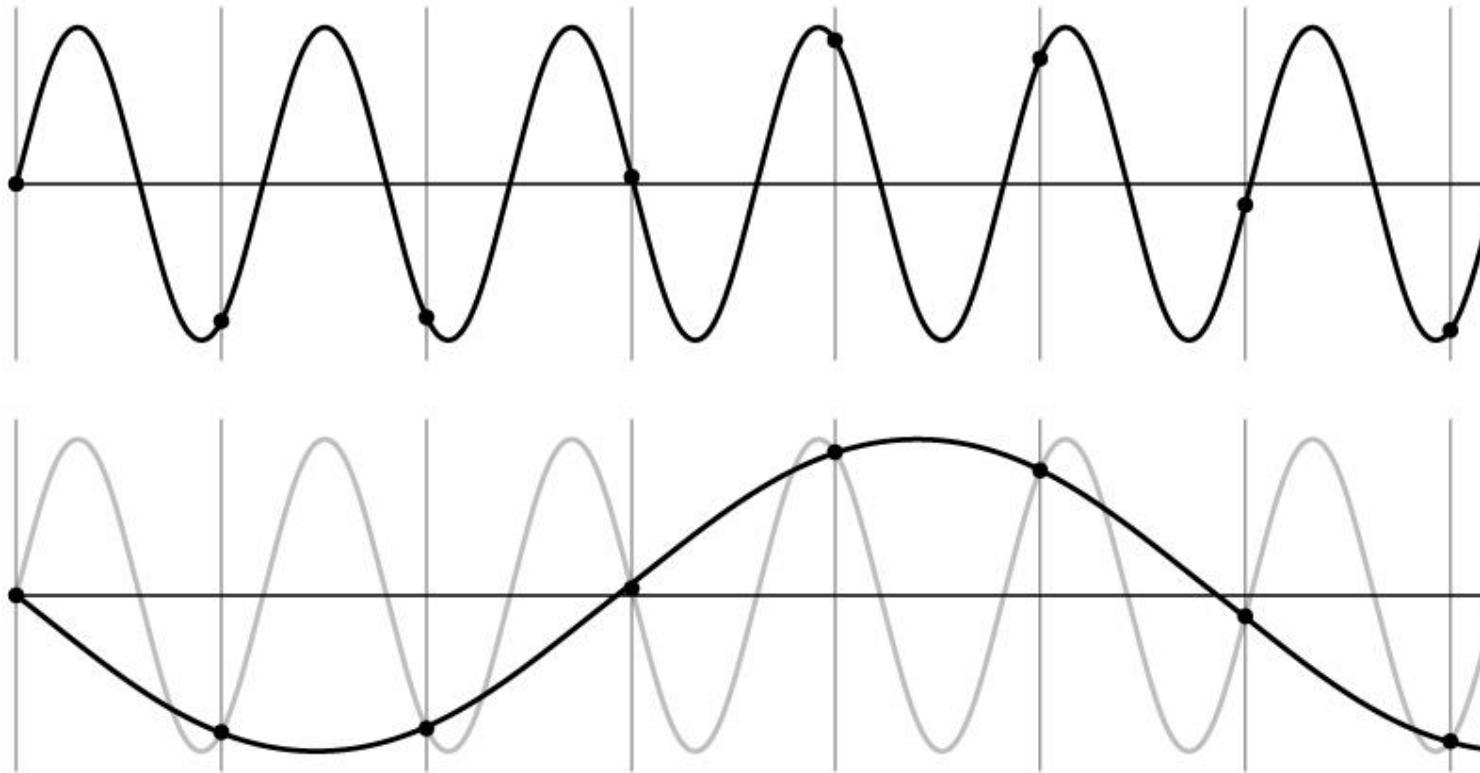
Source: F. Durand

From Continuous to Digital Image



Aliasing Problem

- 1D example (sinewave):



Source: S. Marschner

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”



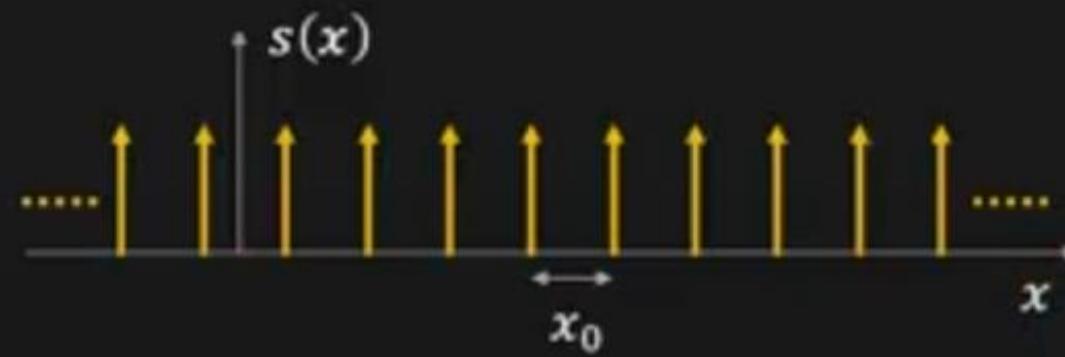
Sampling Theory

Continuous Signal:



Shah Function (Impulse Train):

$$s(x) = \sum_{n=-\infty}^{\infty} \delta(x - nx_0)$$



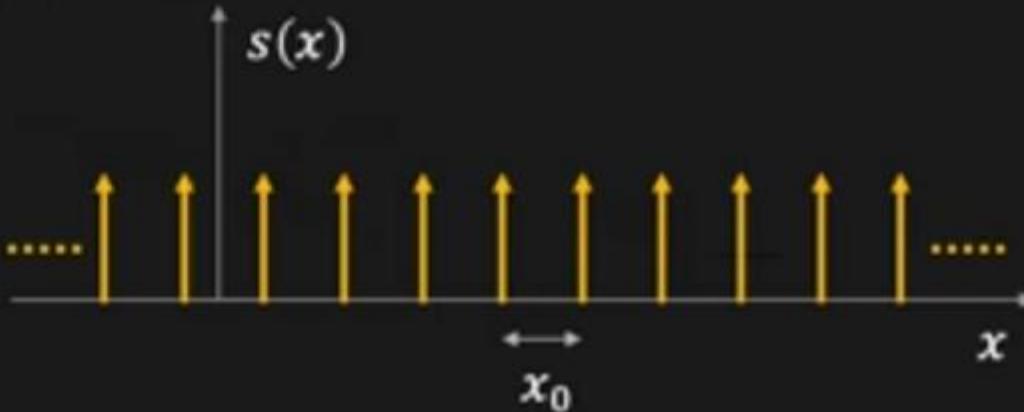
Sampled Function:

$$f_s(x) = f(x)s(x)$$

Shah Function (Impulse Train)

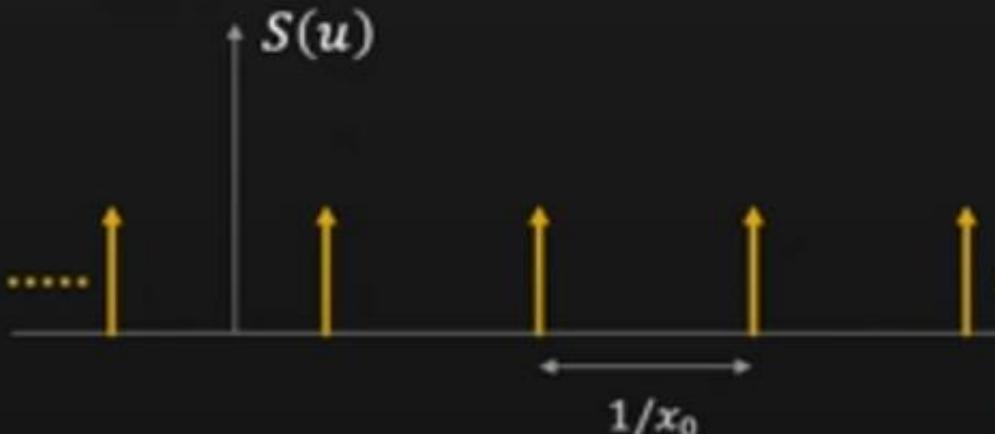
Shah Function (Spatial Domain):

$$s(x) = \sum_{n=-\infty}^{\infty} \delta(x - nx_0)$$



Shah Function (Fourier Domain):

$$S(u) = \frac{1}{x_0} \sum_{n=-\infty}^{\infty} \delta\left(u - \frac{n}{x_0}\right)$$



Fourier Analysis of Sampled Signals

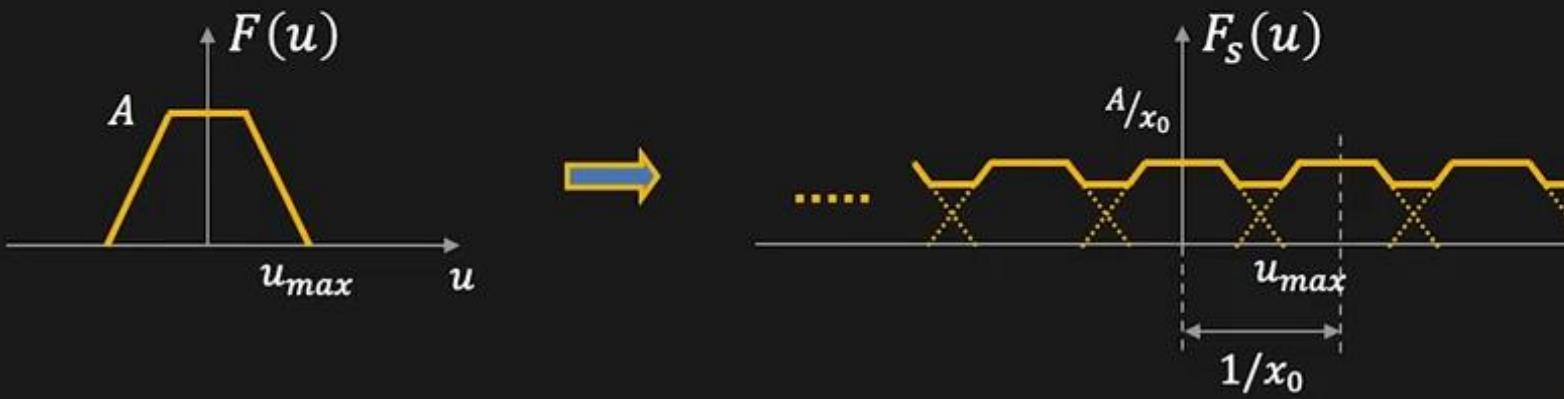
Sampled Signal:

$$f_s(x) = f(x)s(x) = f(x) \sum \delta(x - nx_0)$$

$$F_s(u) = F(u) * S(u) = F(u) * \frac{1}{x_0} \sum \delta(u - n/x_0)$$

If $u_{max} > \frac{1}{2x_0}$

Aliasing



Nyquist Theorem

Can we recover $f(x)$ from $f_s(x)$? In other words,
can we recover $F(u)$ from $F_s(u)$?

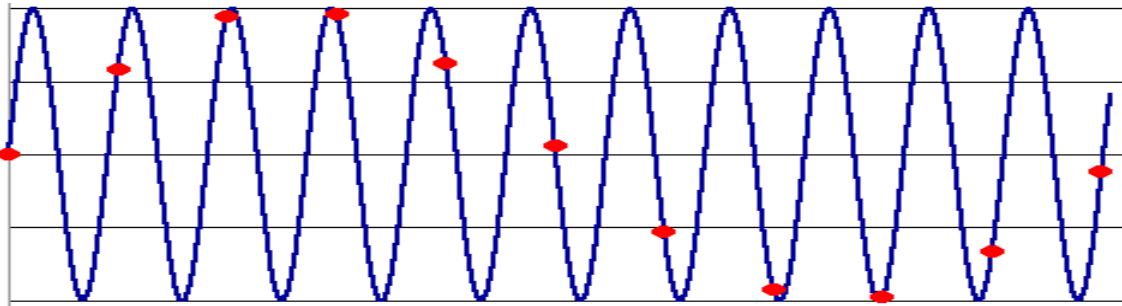
Only if $u_{max} \leq \frac{1}{2x_0}$ (Nyquist Frequency)



$$F(u) = F_s(u)C(u)$$

$$C(u) = \begin{cases} x_0, & |u| < 1/2x_0 \\ 0, & \text{Otherwise} \end{cases}$$

Aliasing



- Occurs when your sampling rate is not high enough to capture the amount of detail in your image
- Can give you the wrong signal/image—an *alias*
- To do sampling right, need to understand the structure of your signal/image
- To avoid aliasing:
 - sampling rate $\geq 2 * \text{max frequency in the image}$
 - said another way: $\geq \text{two samples per cycle}$
 - This minimum sampling rate is called the **Nyquist rate**

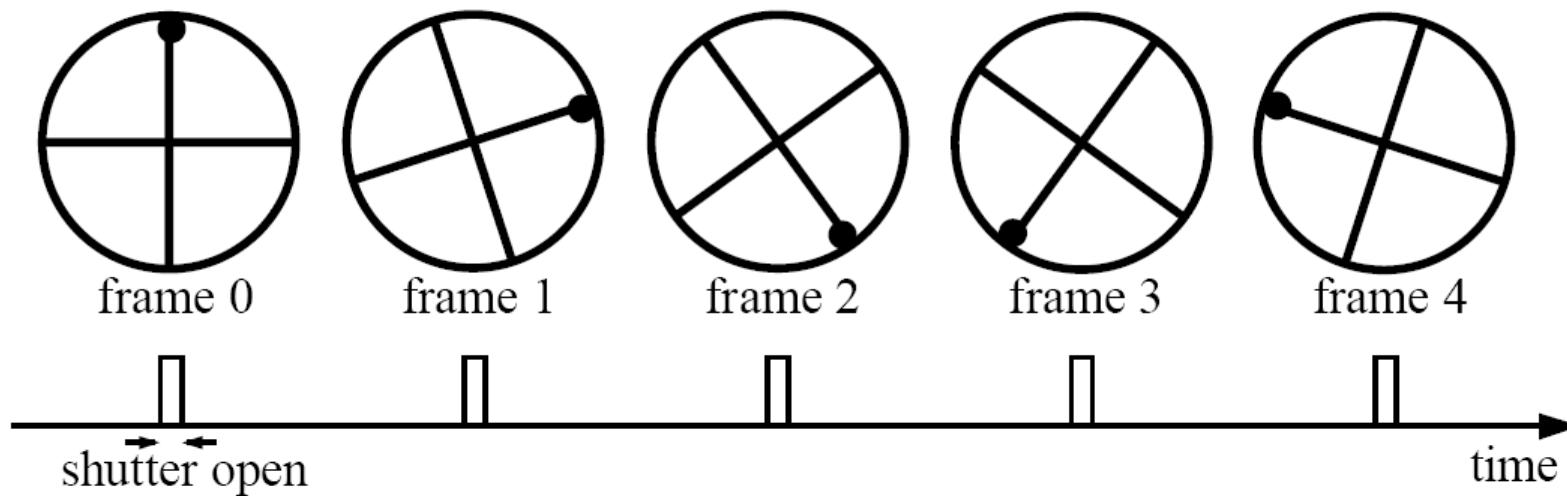
Source: L. Zhang

Wagon Wheel Effect

Imagine a spoked wheel moving to the right (rotating clockwise).

Mark wheel with dot so we can see what's happening.

If camera shutter is only open for a fraction of a frame time (frame time = $1/30$ sec. for video, $1/24$ sec. for film):



Without dot, wheel appears to be rotating slowly backwards!
(counterclockwise)

(See http://www.michaelbach.de/ot/mot_wagonWheel/index.html)

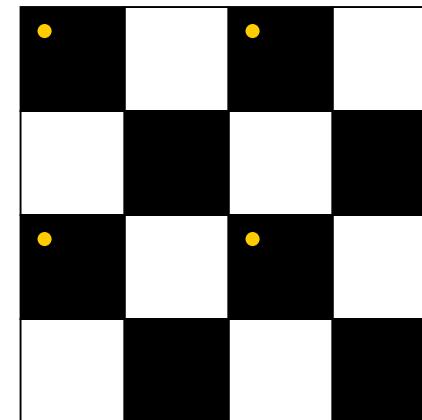
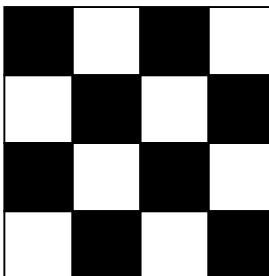
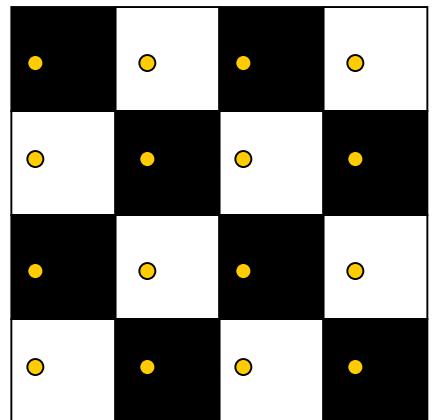
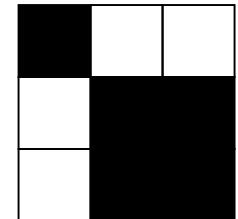
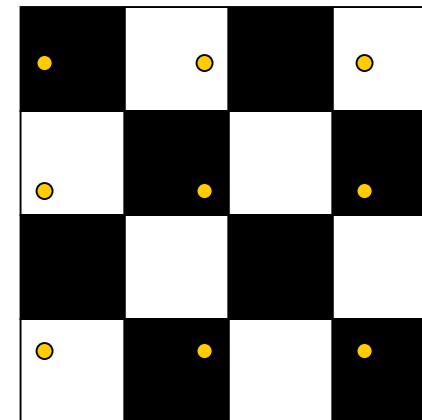
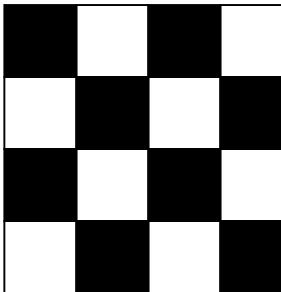
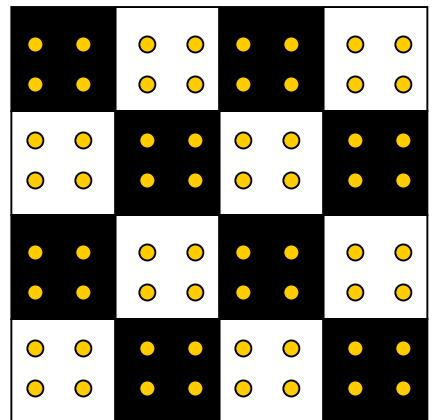
Source: L. Zhang

Wagon-Wheel Effect



https://www.youtube.com/watch?v=QOwzkND_ooU

Sampling an Image



Examples of GOOD sampling

Examples of BAD sampling -> Aliasing

Example



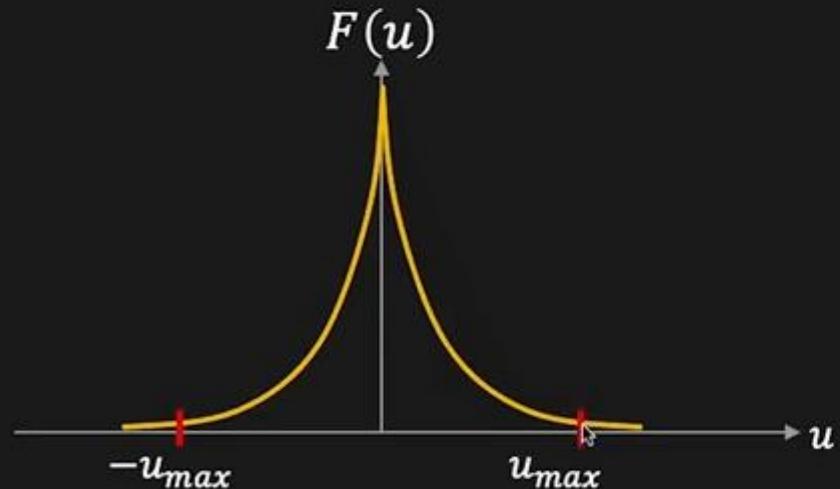
"Well sampled" image



"Under sampled" image
(visible **aliasing** artifacts)

Aliasing in Digital Imaging

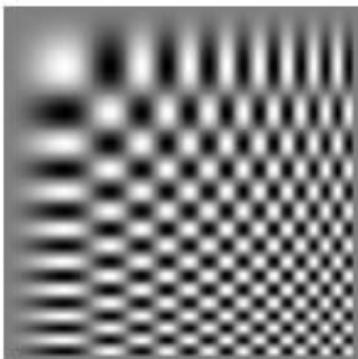
Aliasing occurs when imaging a scene (signal) that has frequencies above the image sensor's Nyquist Frequency



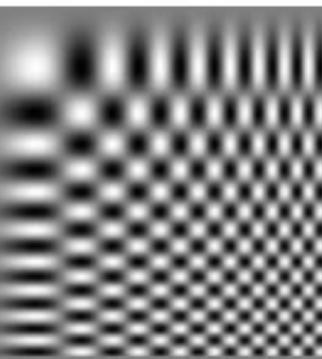
Aliasing artifacts usually occur in the form of Moiré patterns

Anti-Aliasing

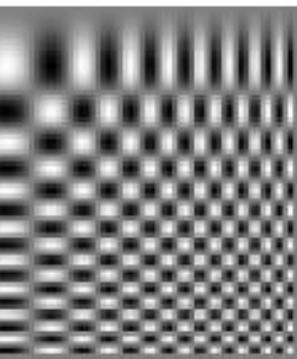
256x256



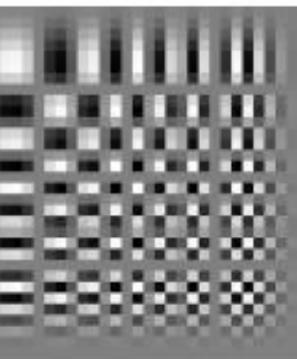
128x128



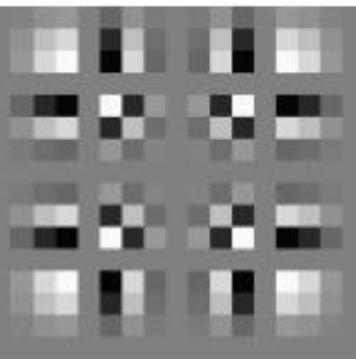
64x64



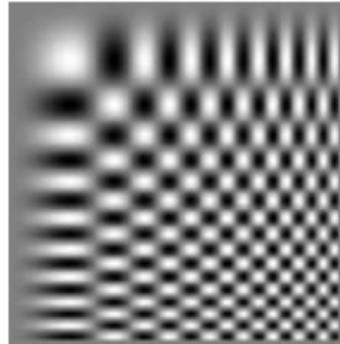
32x32



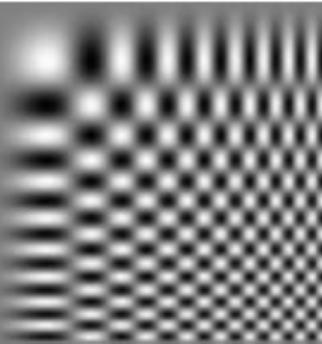
16x16



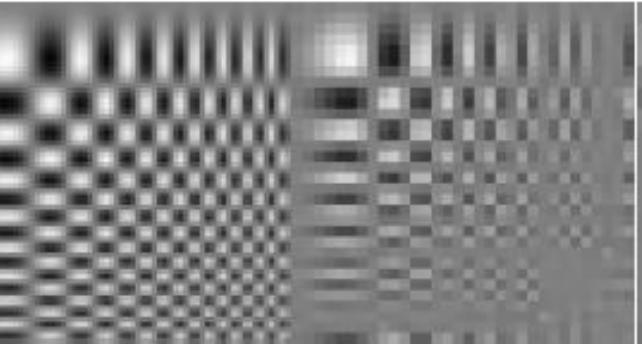
256x256



128x128



64x64



32x32



16x16



Forsyth and Ponce 2002

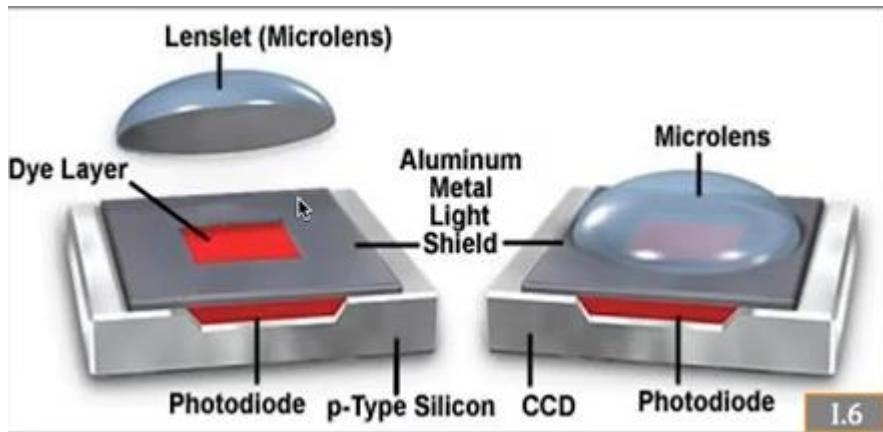
Dr. Sander Ali Khowaja

Minimizing the Effects of Aliasing

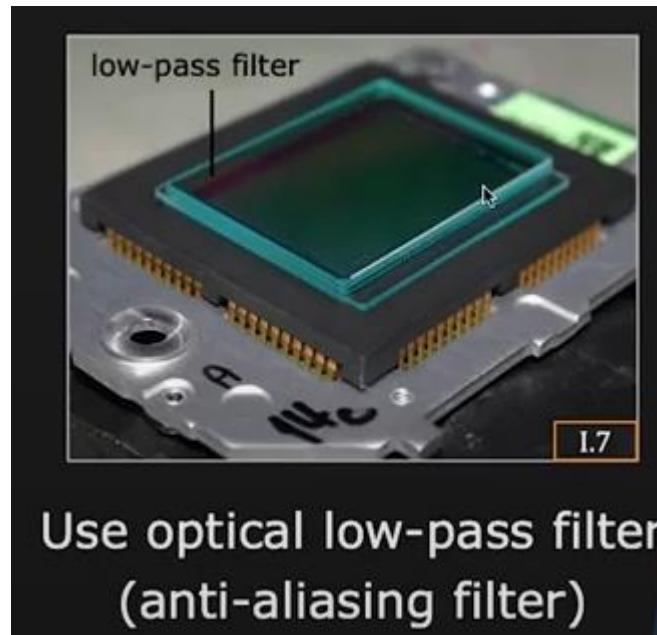
Band Limit: Clip the signal above the Nyquist Frequency.

Effectively, “blur” the scene before sampling.

Sensors use two strategies

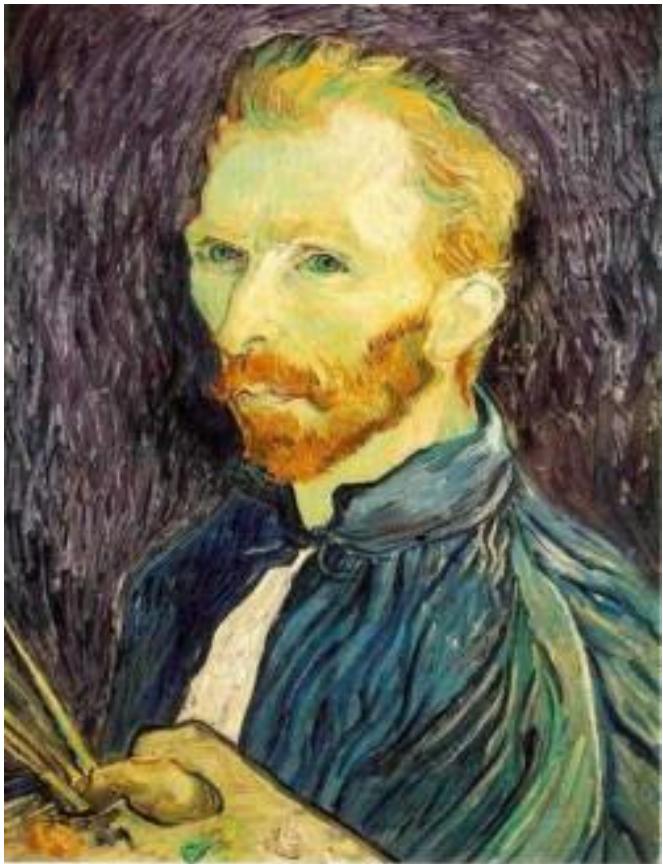


Pixels are area-samplers
(box-averaging filter)



Use optical low-pass filter
(anti-aliasing filter)

Gaussian (Low-Pass) Pre-Filtering



Gaussian 1/2



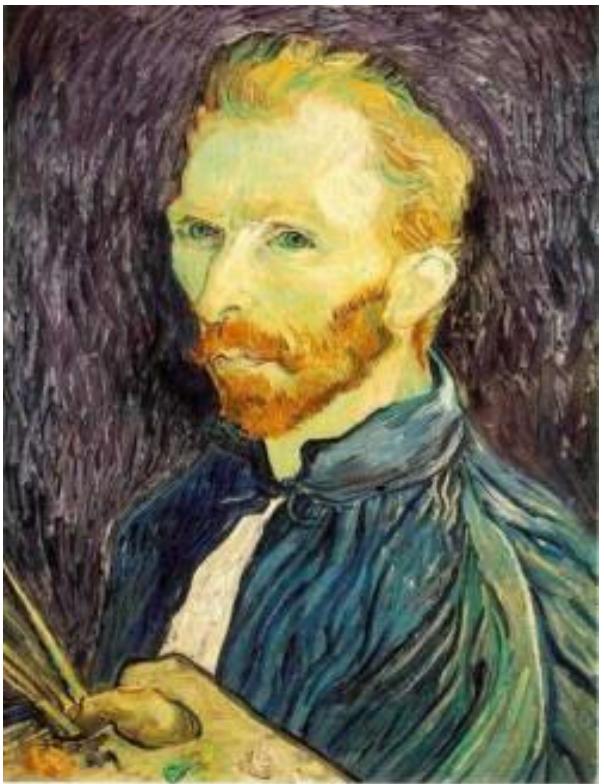
G 1/4



G 1/8

- Solution: filter the image, *then* subsample

Subsampling with Gaussian Pre-Filtering



Gaussian 1/2



G 1/4

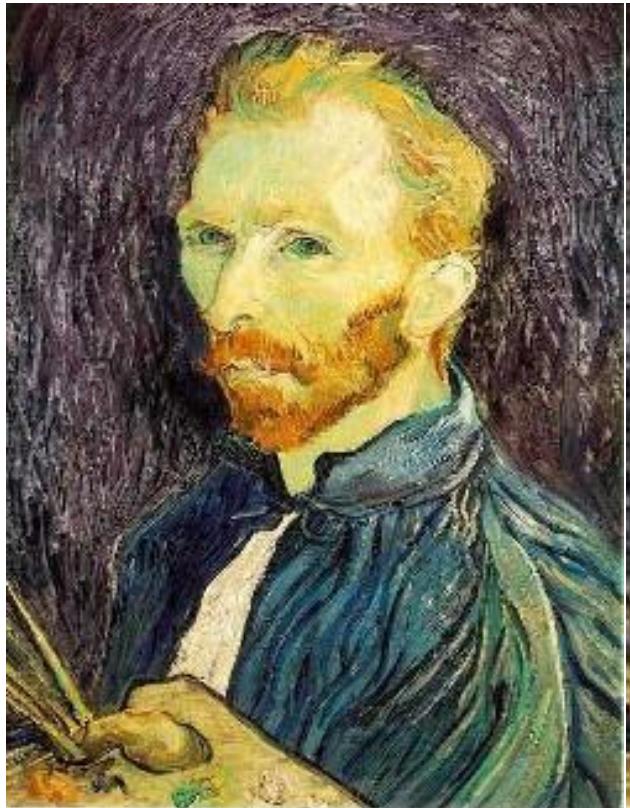


G 1/8

- Solution: filter the image, *then* subsample

Source: S. Seitz

Compare With



1/2

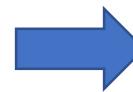


1/4 (2x zoom)



1/8 (4x zoom)

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



Human Vision: Fovea and Periphery

L C C O
A I M A
C I X T A T G R
C F I F T U C O N
I I N O N T

Some properties of image encoding like blurring, color representation set the stage for what is available to the neural system



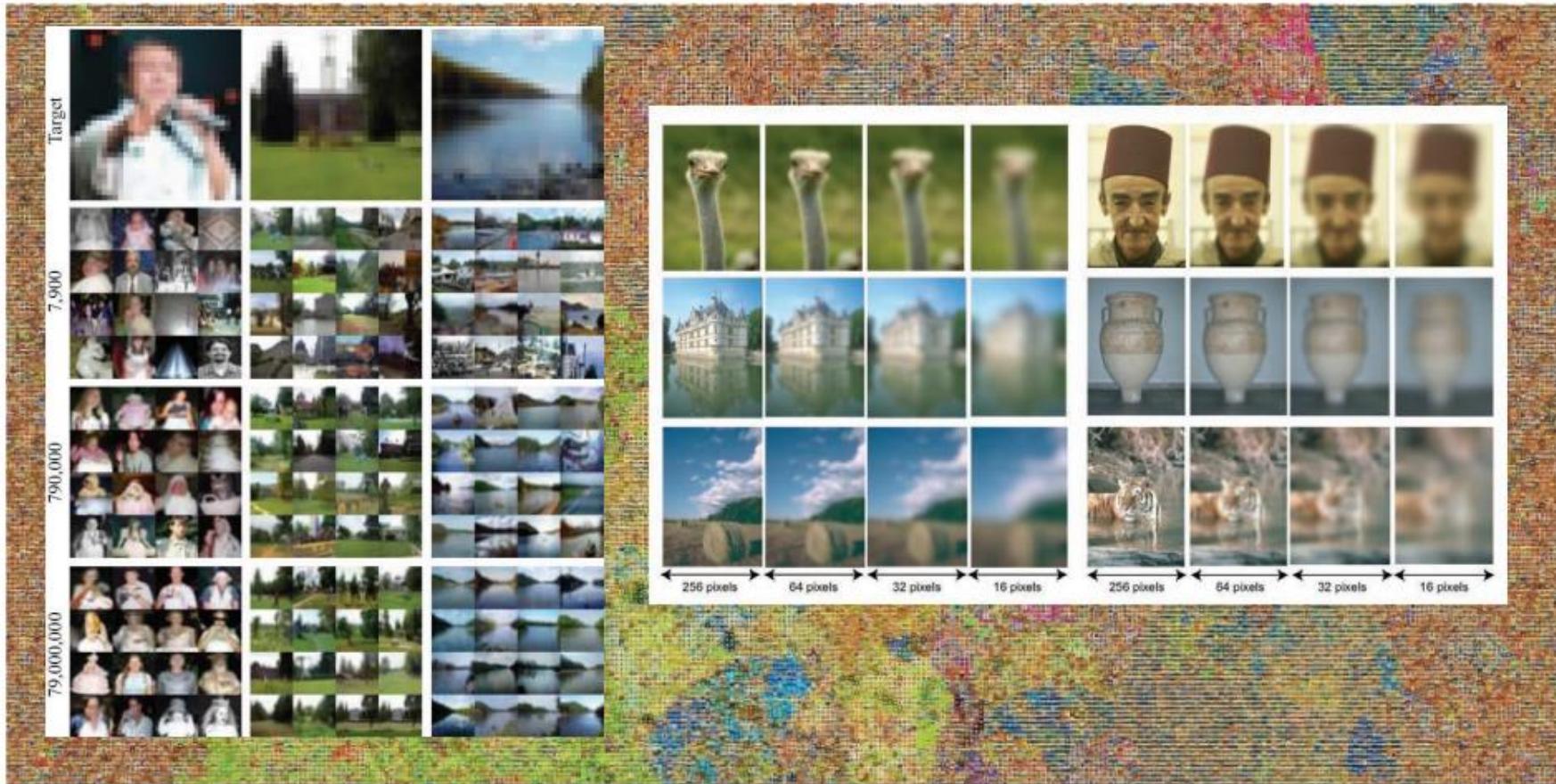
Camera



Human: Acuity decreases with eccentricity

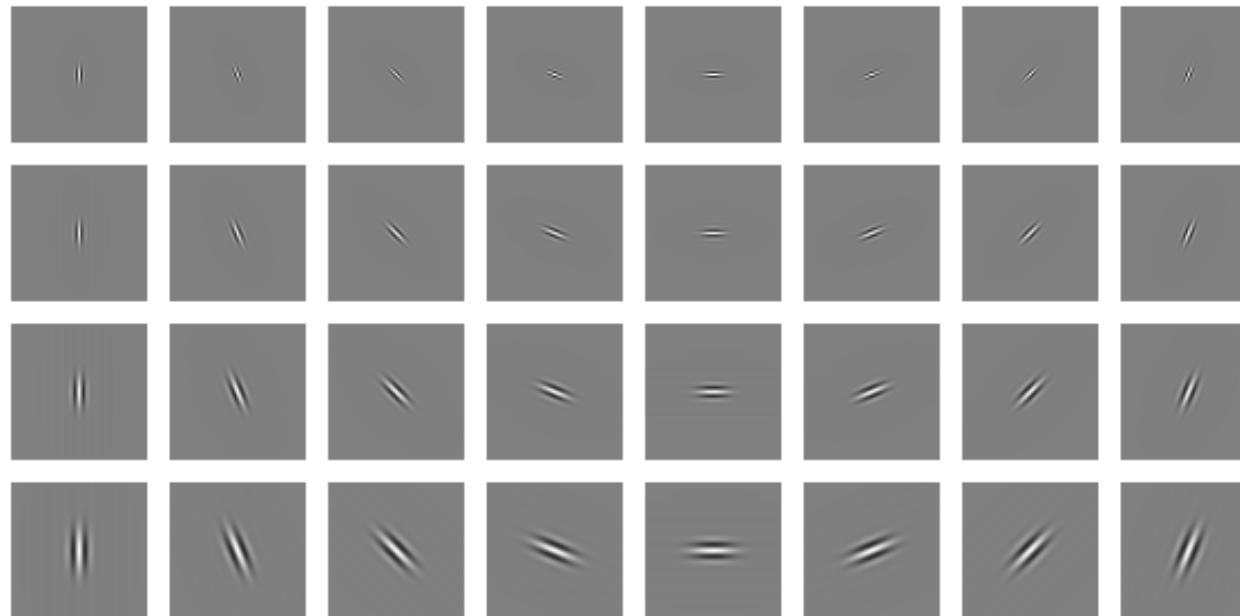
Human Vision: Receptive Field size scale with eccentricity

80 millions tiny images



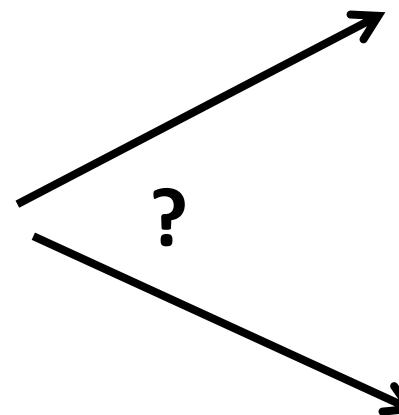
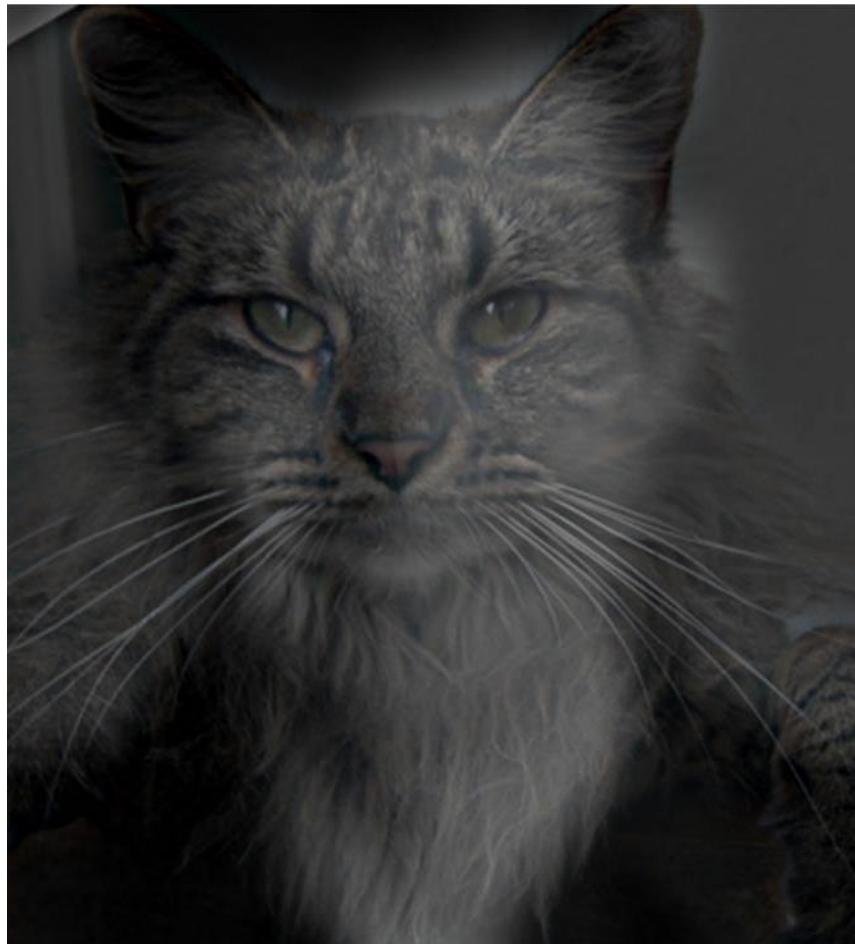
Clues from Human Perception

- 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



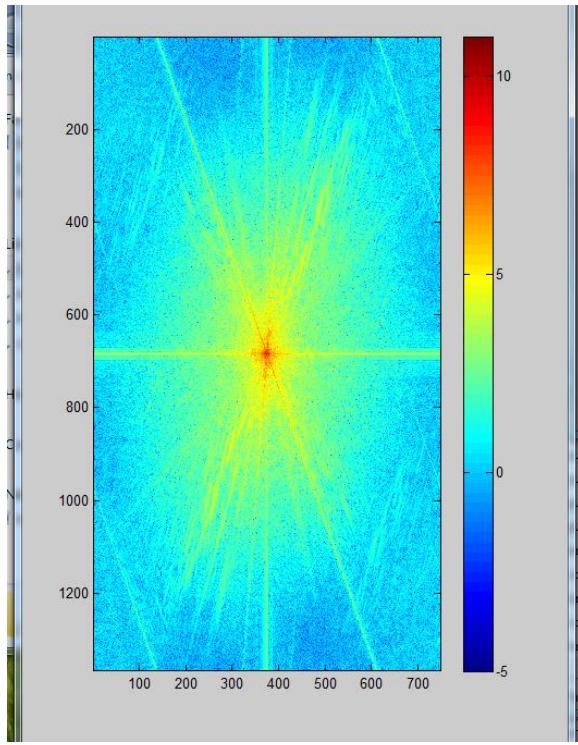
Early Visual Processing: Multi-scale edge and blob filters

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

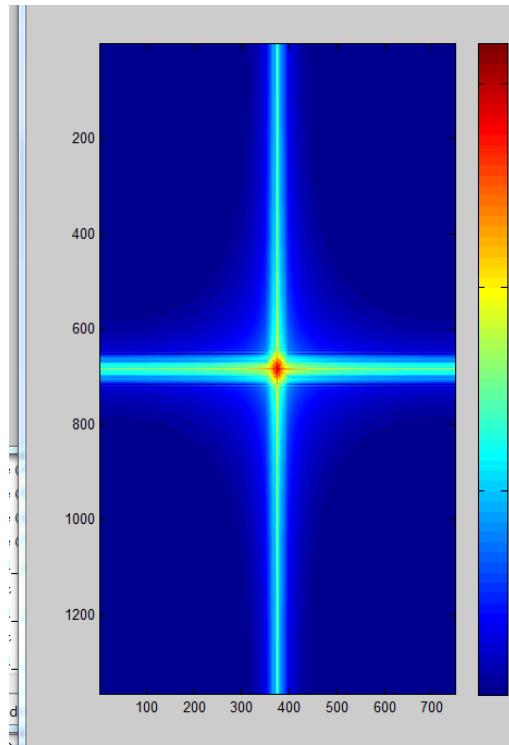


Hybrid Image in FFT

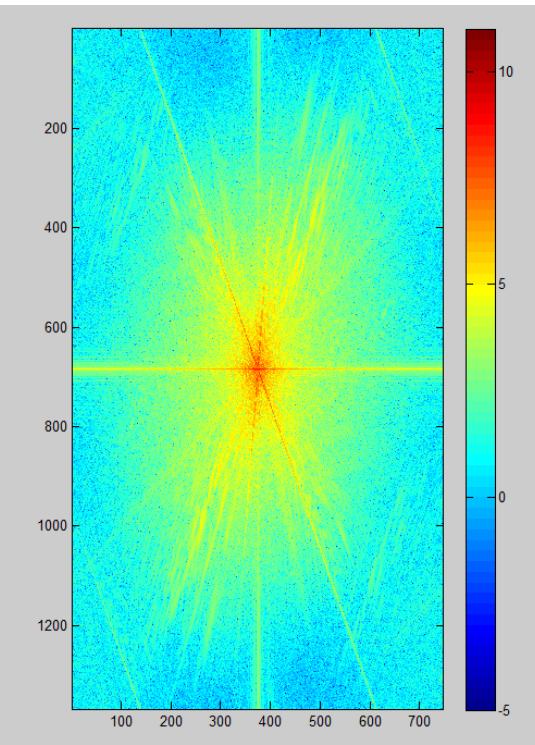
Hybrid Image



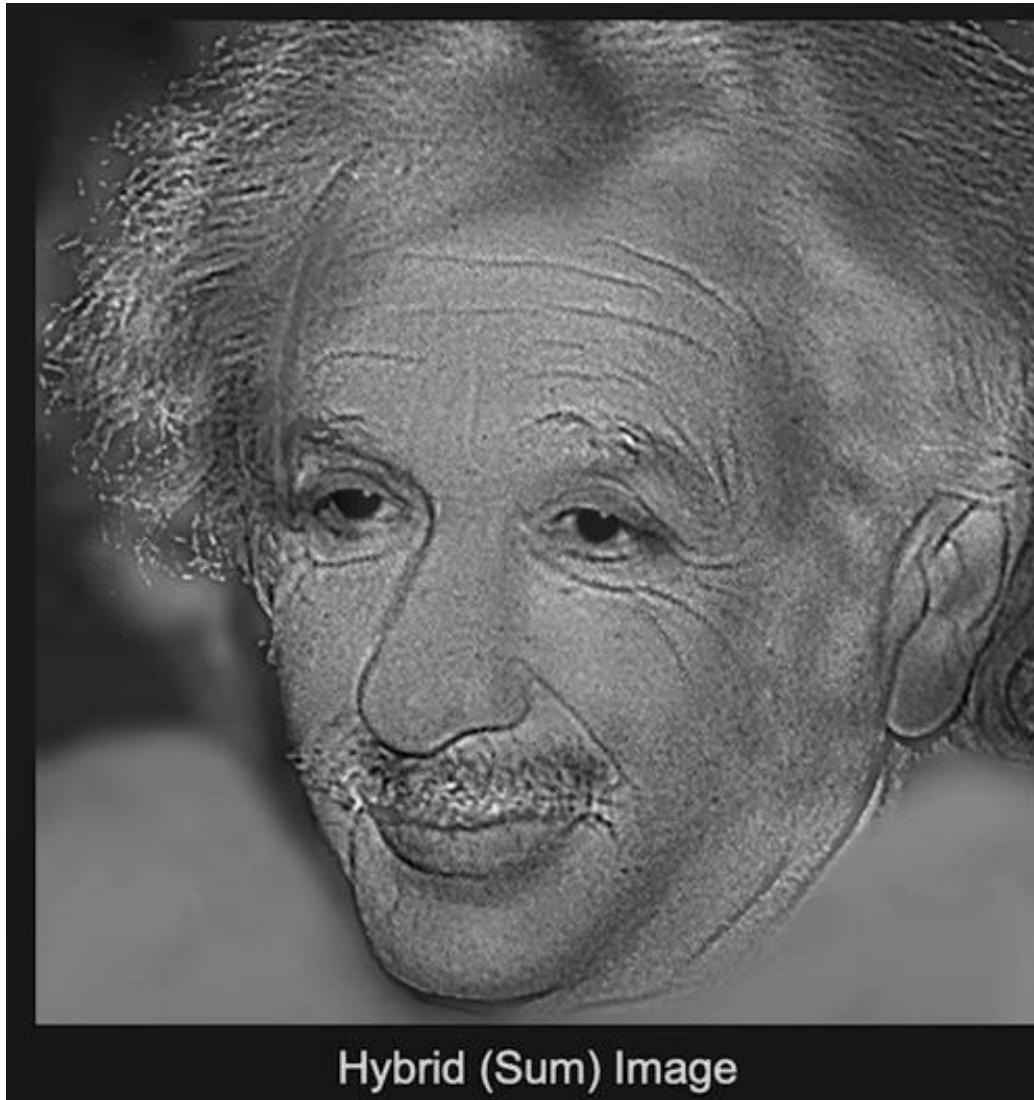
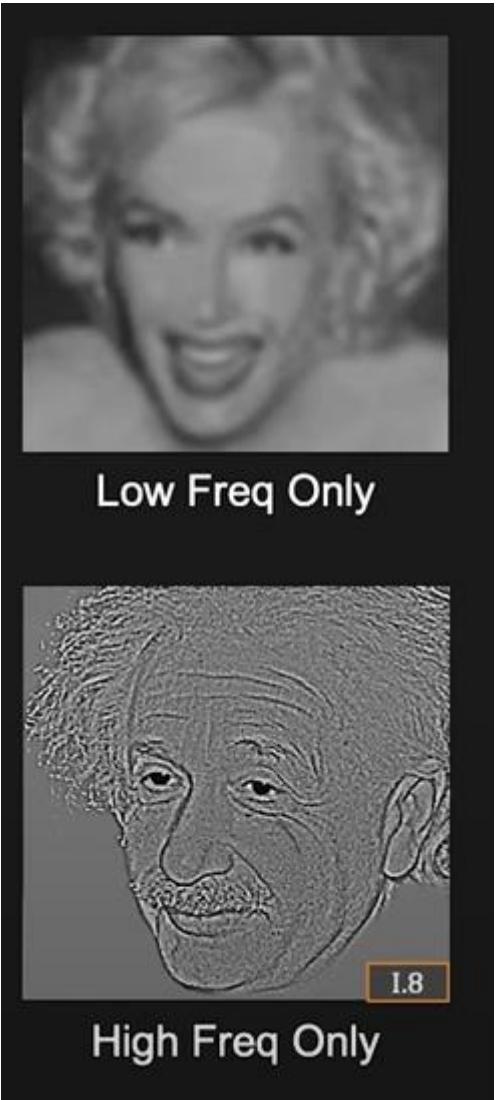
Low-passed Image



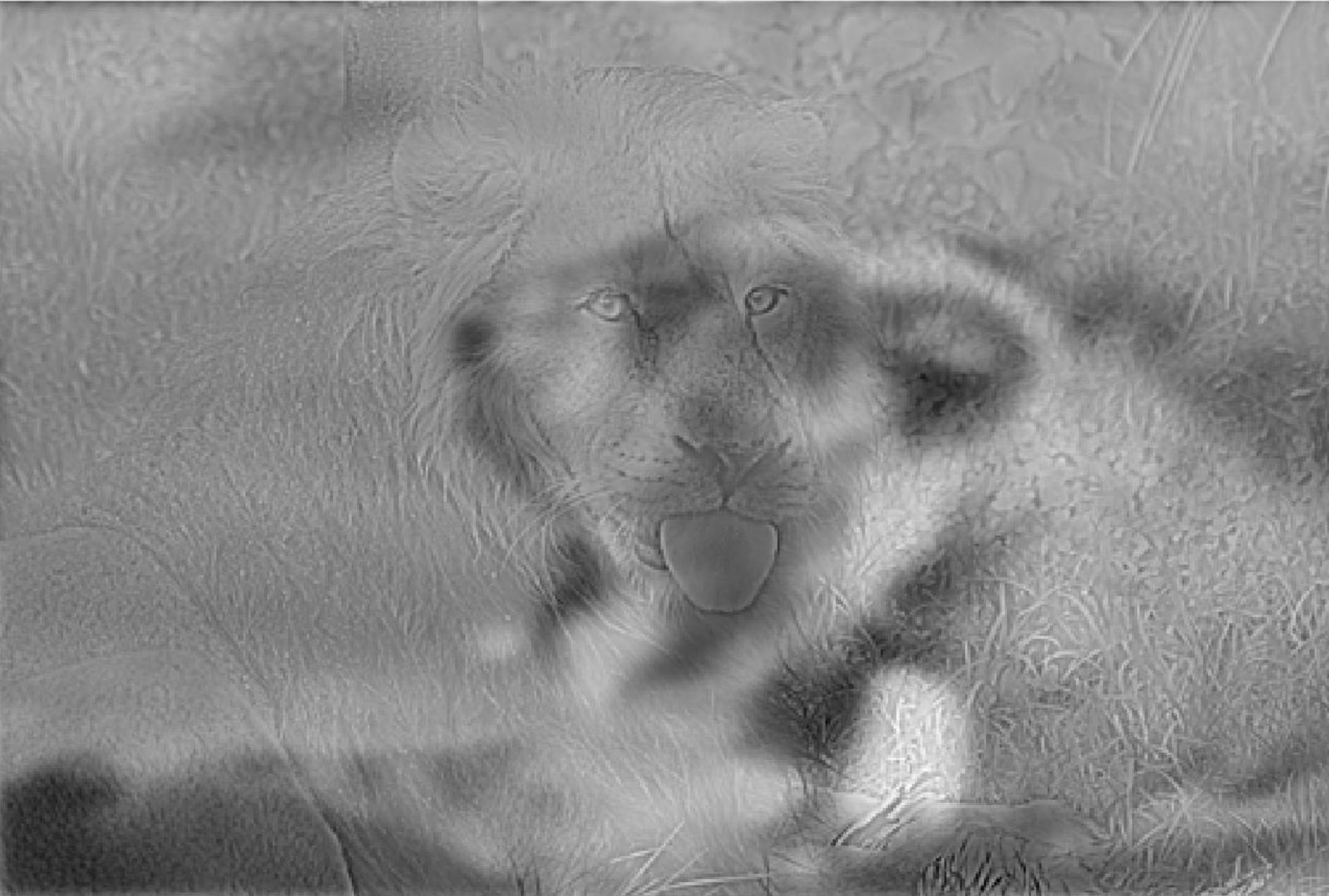
High-passed Image



Hybrid Images



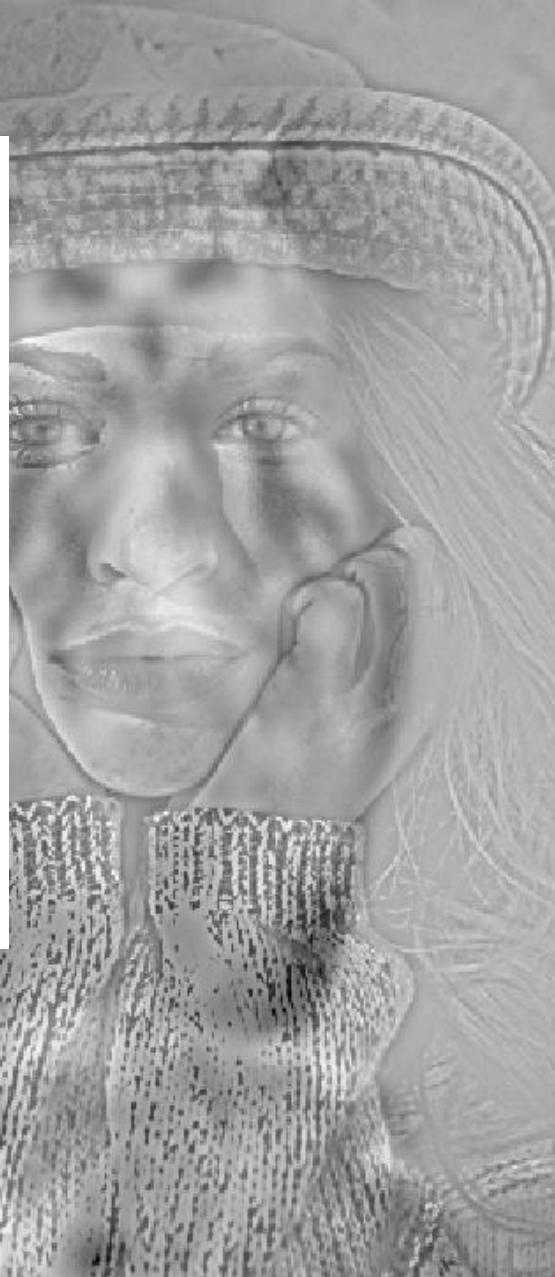
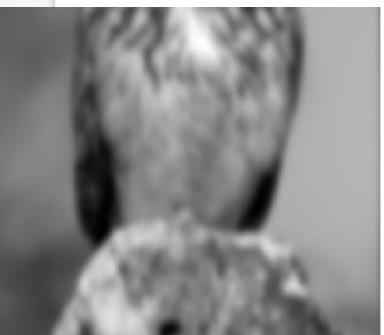
Hybrid Image Example



Hybrid Image Example



```
1 im1 = imread('mm1.jpg');
2 im2 = imread('ae1.jpg');
3 im1 = imresize(im1,[347 317]);
4 im11 = histeq(im2double(rgb2gray(im1)));
5 im12 = im2double(rgb2gray(im2));
6
7 sigma = 3;
8 gaussian_dimension = 3*sigma*2+1;
9
10 im11_hpf = highPassFilter(im11,gaussian_dimension);
11
12 imshow(im11_hpf,[])
13
14 sigma = 2;
15 gaussian_dimension=3*sigma*2+1;
16 im12_lpf = lowPassFilter(im12,gaussian_dimension);
17
18 figure, imshow(im12_lpf,[])
19
20 himage = im11_hpf + im12_lpf;
21
22 figure, imshow(himage,[])
```



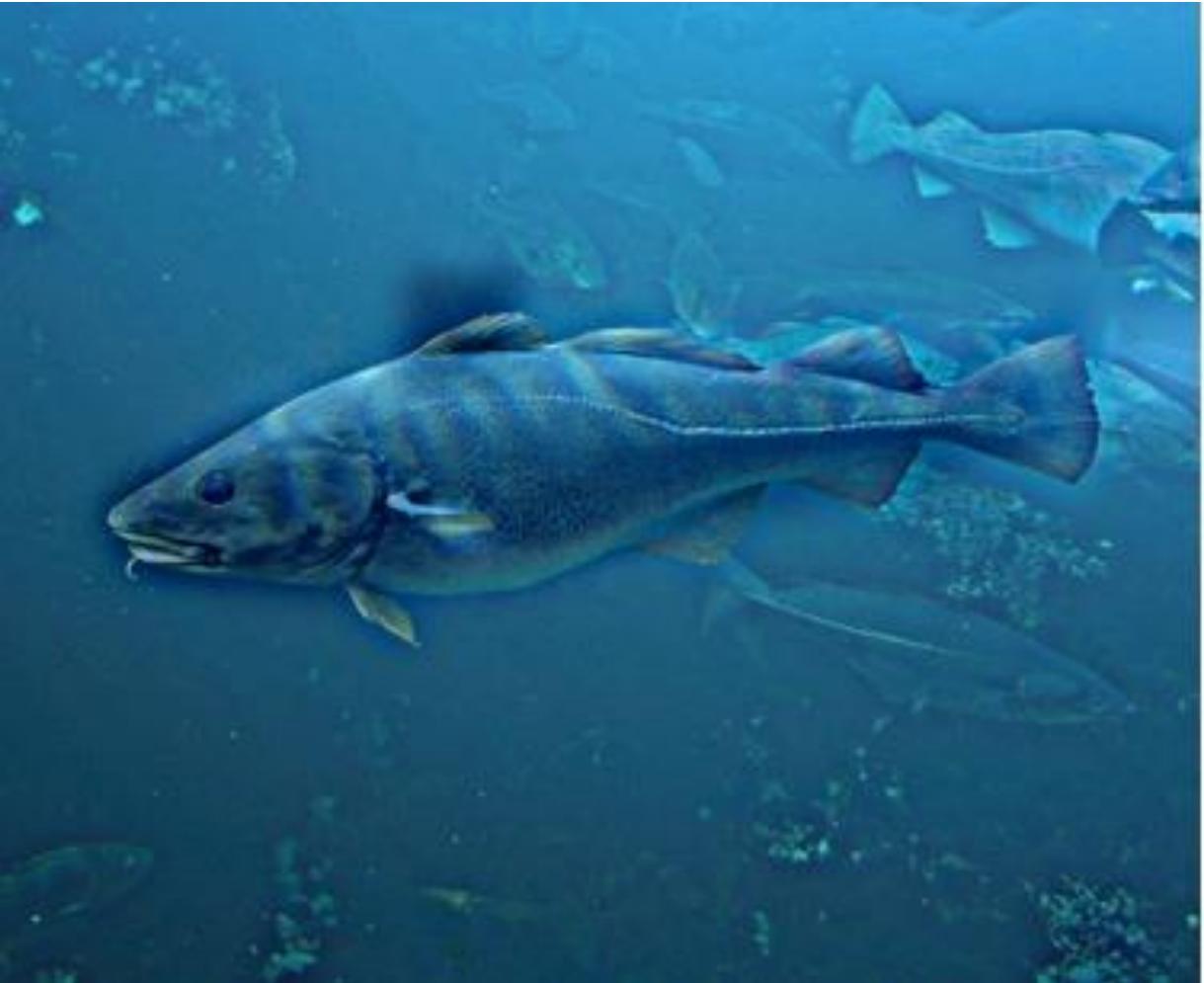
Some Examples of Scale-based Hybrid Images



Some Examples of Scale-based Hybrid Images



Some Examples of Scale-based Hybrid Images



Some Examples of Scale-based Hybrid Image



Some Examples of Scale-based Hybrid Image

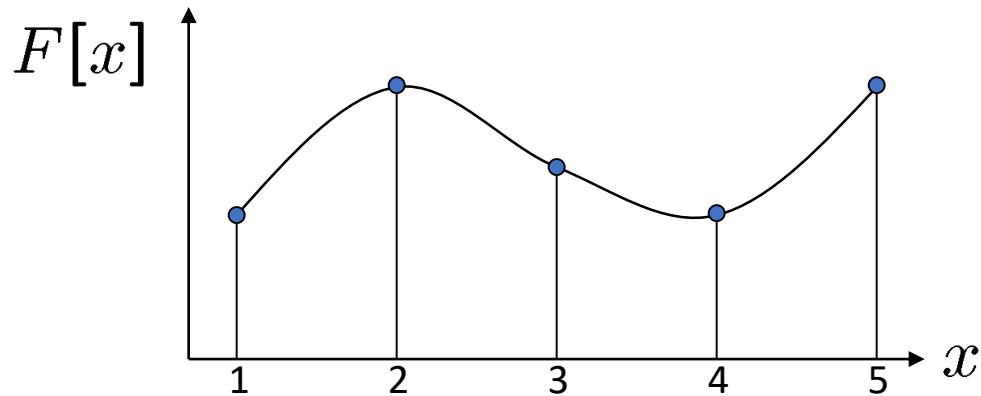


Upsampling

- This image is too small for this screen:
- How can we make it 10 times as big?
- Simplest approach:
 - repeat each row
 - and column 10 times
- (“Nearest neighbor interpolation”)



Image Interpolation



$d = 1$ in this example

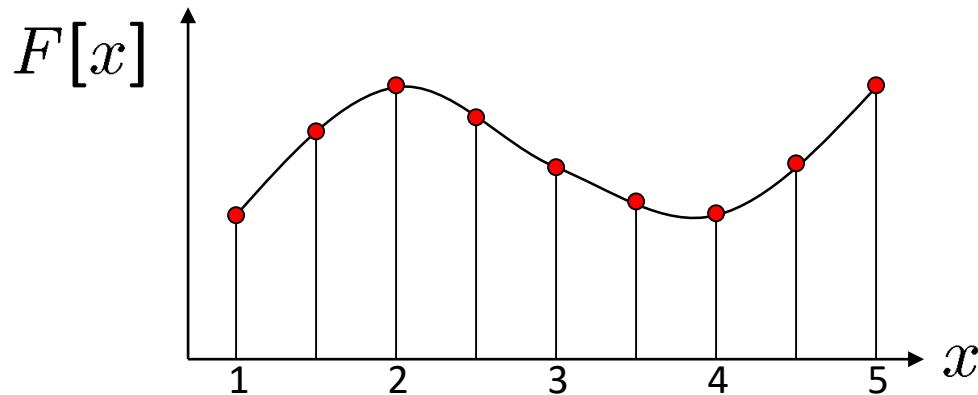
Recall how a digital image is formed

$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$

- It is a discrete point-sampling of a continuous function
- If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale

Adapted from: S. Seitz

Image Interpolation



$d = 1$ in this example

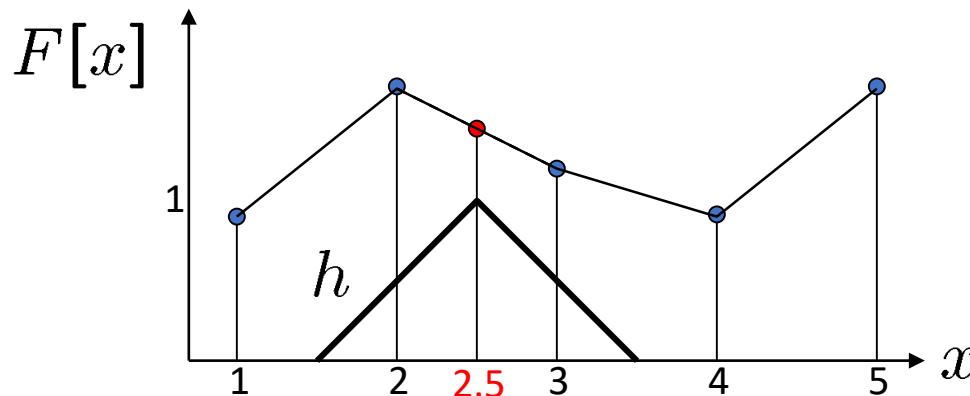
Recall how a digital image is formed

$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$

- It is a discrete point-sampling of a continuous function
- If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale

Adapted from: S. Seitz

Image Interpolation

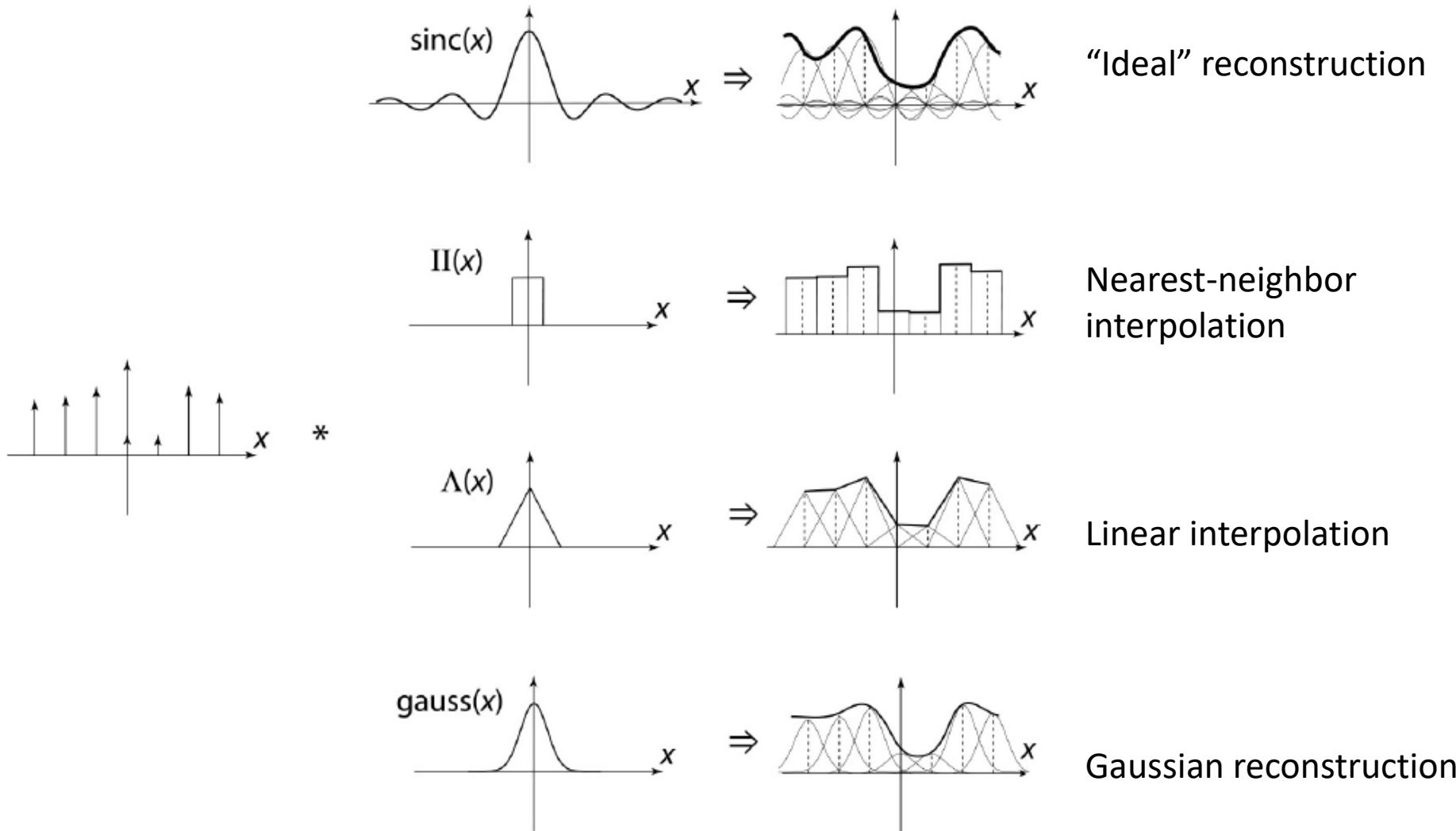


$d = 1$ in this example

- What if we don't know f ?
 - Guess an approximation: \tilde{f}
 - Can be done in a principled way: filtering
 - Convert F to a continuous function:
$$f_F(x) = F\left(\frac{x}{d}\right)$$
 when $\frac{x}{d}$ is an integer, 0 otherwise
 - Reconstruct by convolution with a *reconstruction filter*, h
$$\tilde{f} = h * f_F$$

Adapted from: S. Seitz

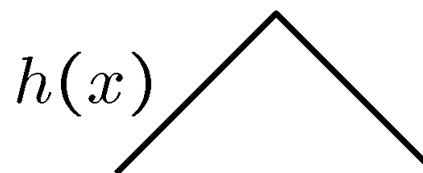
Image Interpolation



Source: B. Curless

Reconstruction Filters

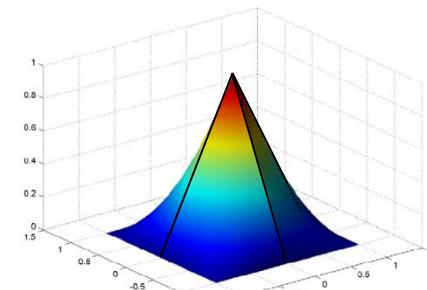
- What does the 2D version of this hat function look like?



performs
linear interpolation



(tent function) performs
bilinear interpolation

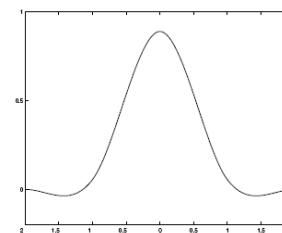


Often implemented without cross-correlation

- E.g., http://en.wikipedia.org/wiki/Bilinear_interpolation

Better filters give better resampled images

- **Bicubic** is common choice



Cubic reconstruction filter

$$r(x) = \begin{cases} (12 - 9B - 6C)|x|^3 + (-18 + 12B + 6C)|x|^2 + (6 - 2B) & |x| < 1 \\ ((-B - 6C)|x|^3 + (6B + 30C)|x|^2 + (-12B - 48C)|x| + (8B + 24C)) & 1 \leq |x| < 2 \\ 0 & \text{otherwise} \end{cases}$$

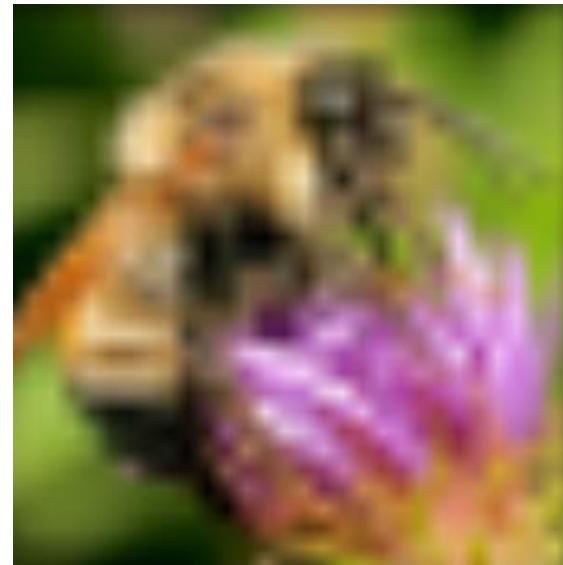


Image Interpolation

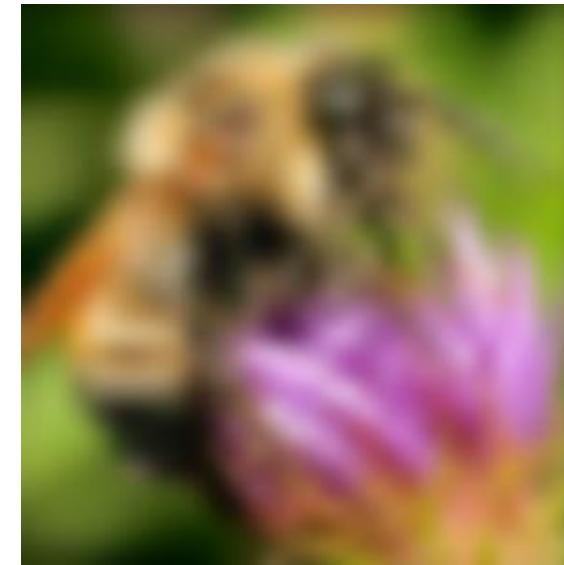
Original image:  x 10



Nearest-neighbor interpolation



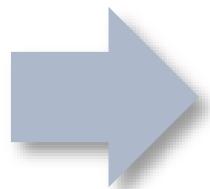
Bilinear interpolation



Bicubic interpolation

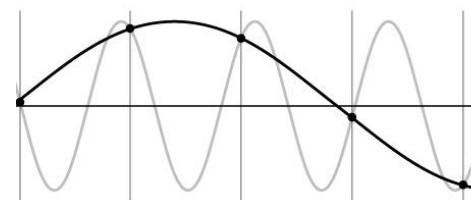
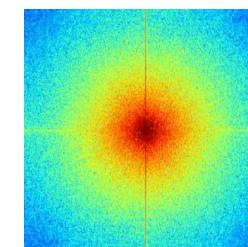
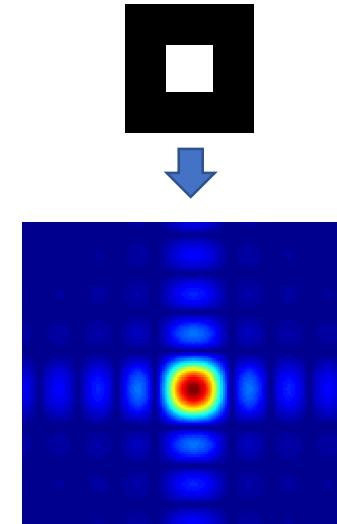
Image Interpolation

Also used for *resampling*



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 ($N \log N$ vs. N^2 for auto-correlation)
- Images are mostly smooth
 - Basis for compression
- Remember to low-pass before sampling



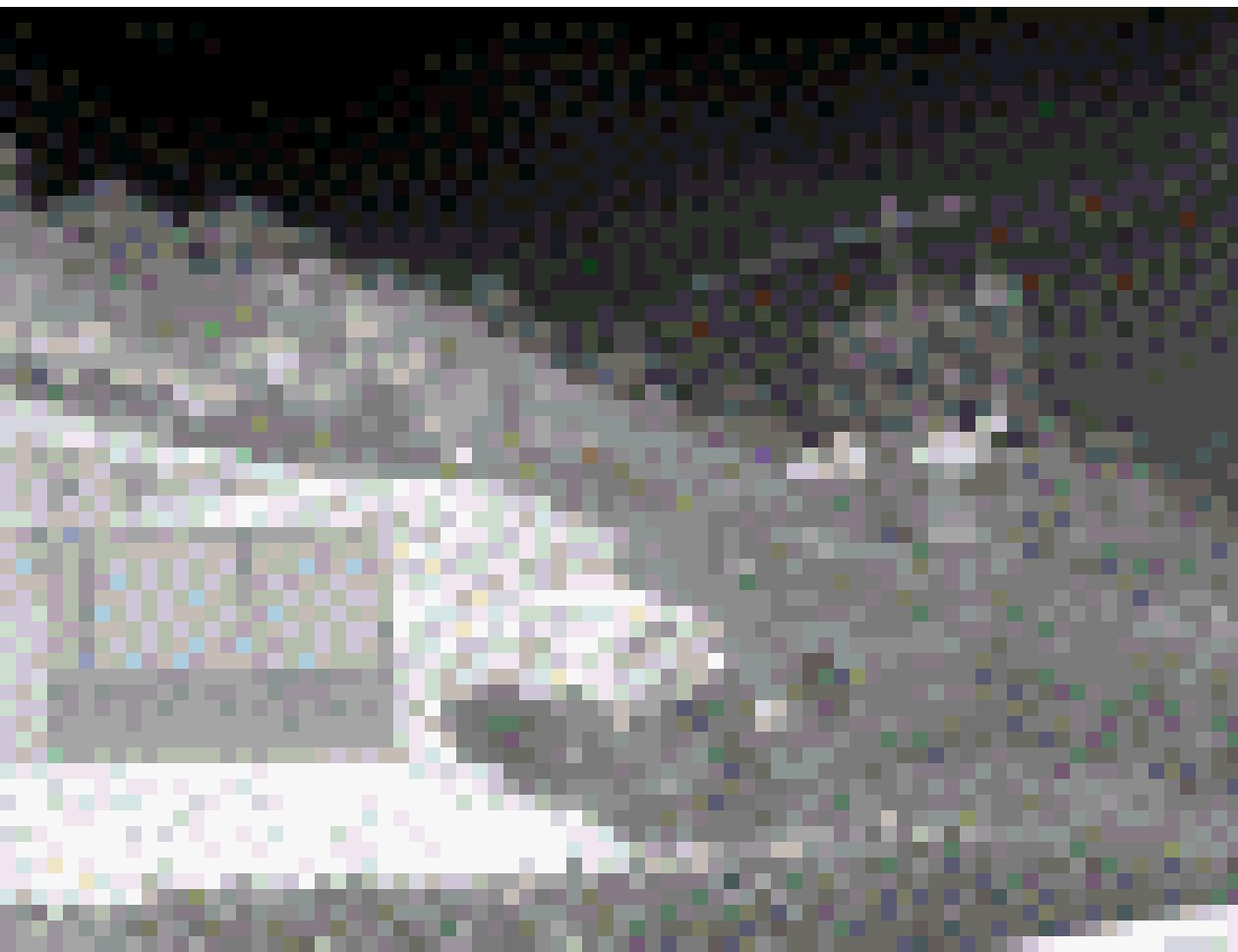
Some Examples



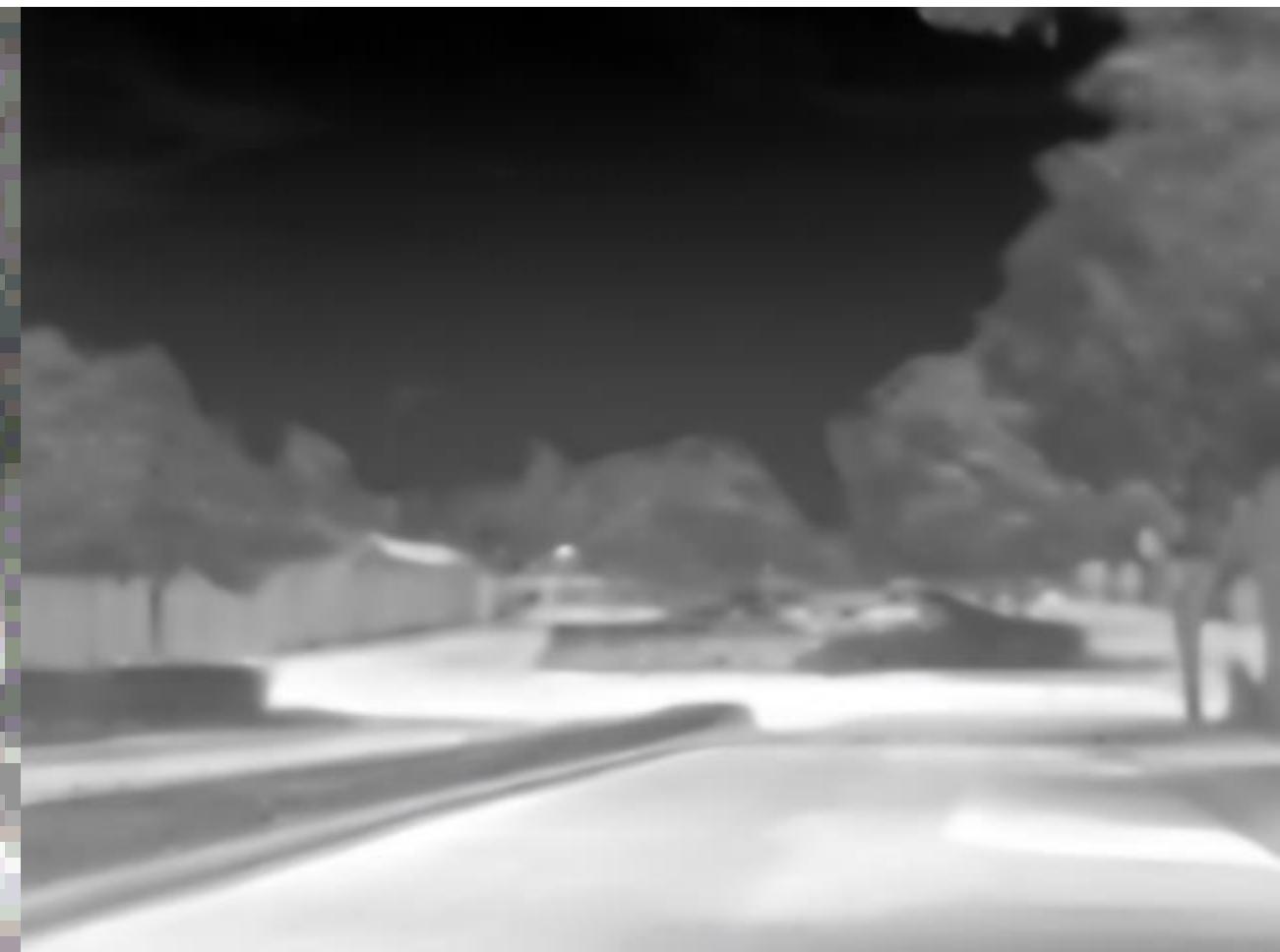
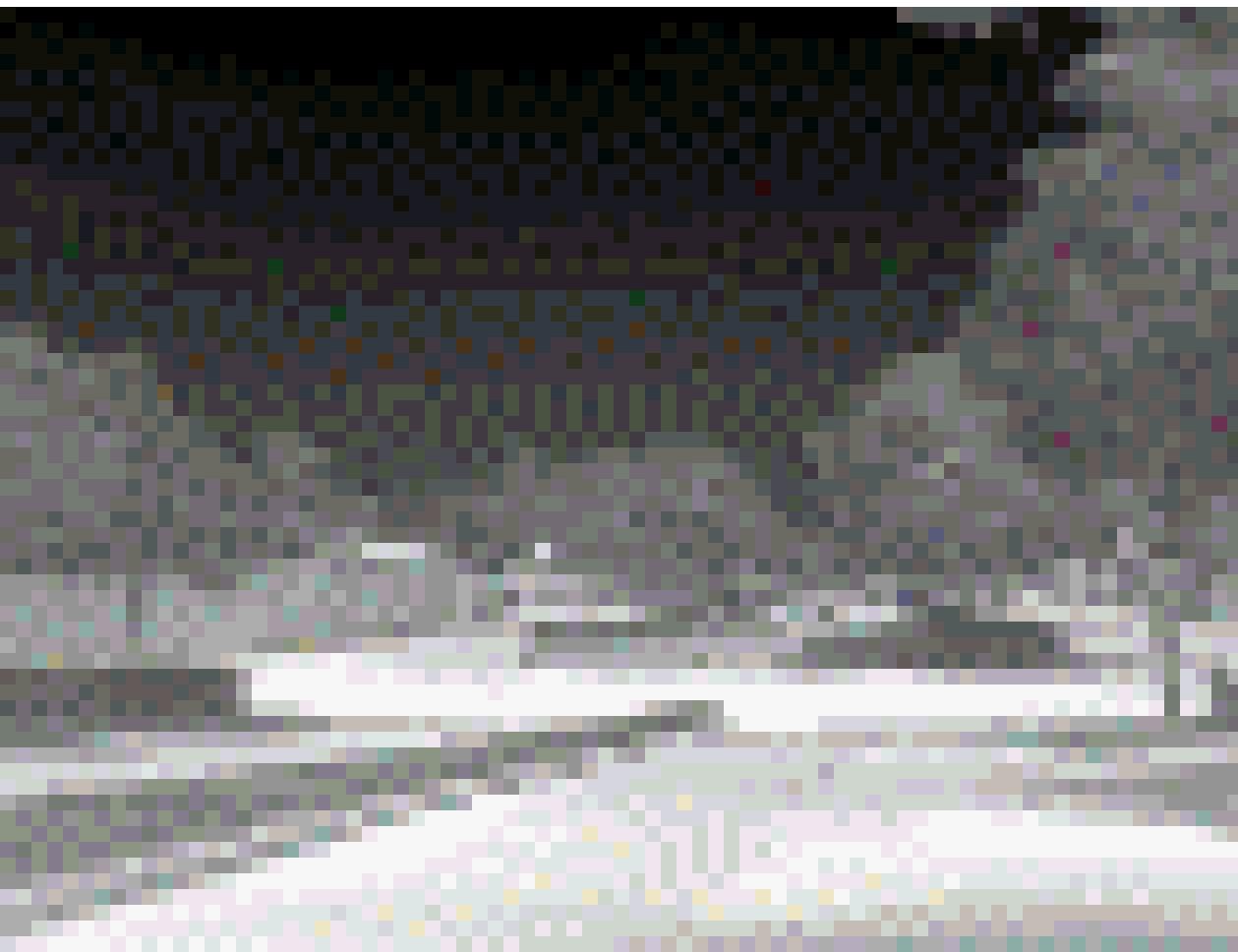
x8
→

A blue arrow pointing from the small image to the large one, indicating an 8x magnification factor.

Some Examples



Some Examples



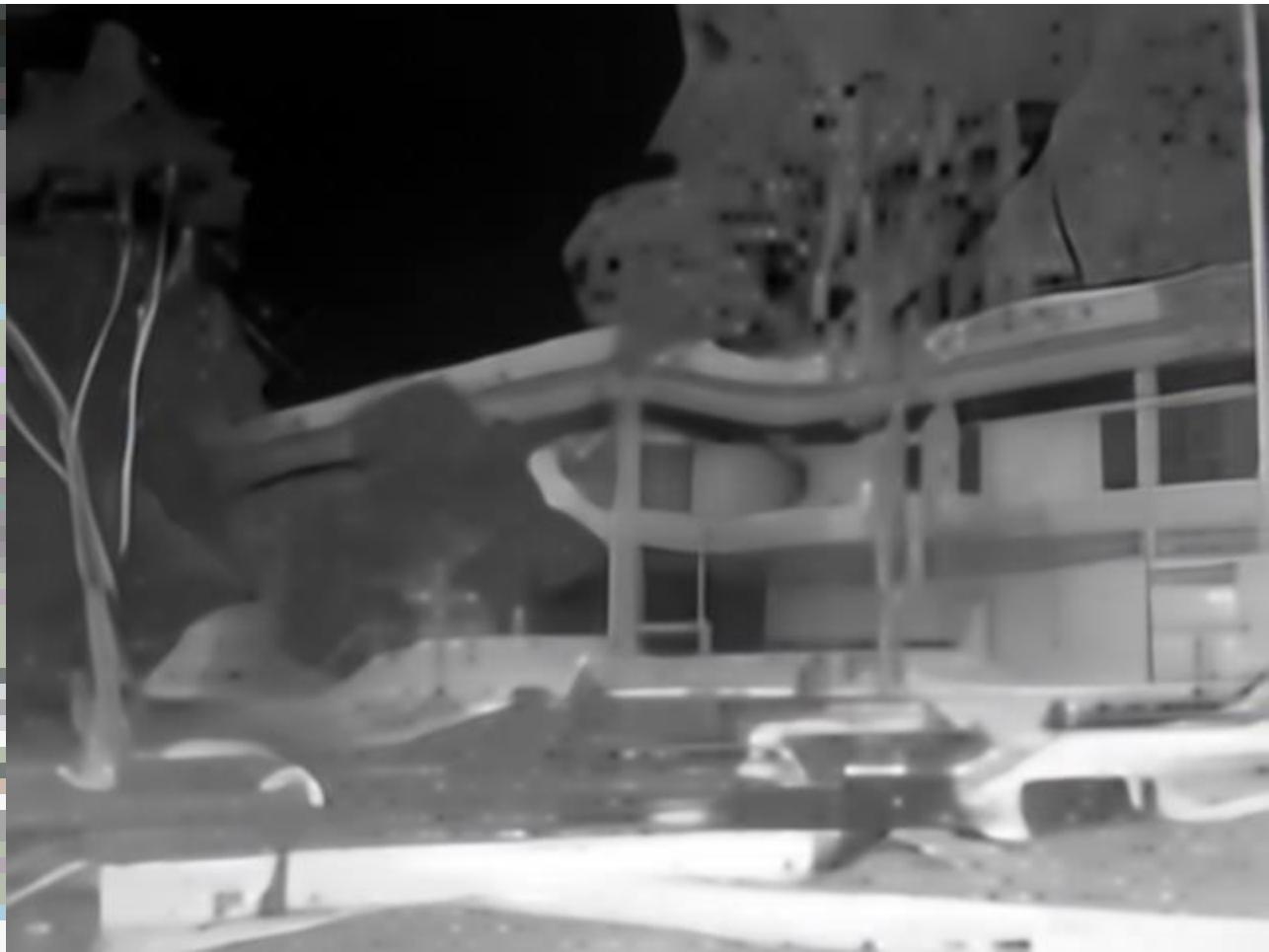
Some Examples



Some Examples



Some Examples



Python code for image super resolution using EDSR

```
• import cv2  
• from cv2 import dnn_superres  
• import os  
• from os import listdir  
• import glob  
• sr = dnn_superres.DnnSuperResImpl_create()  
• path = "EDSR_x2.pb"  
• sr.readModel(path)  
• sr.setModel("edsr",2)  
• img = cv2.imread("image path")  
• result = sr.upsample(img)  
• cv2.imwrite(filename,result)
```



- MATLAB provides an example of super resolution using deep learning method VDSR
- Follow the steps in the link below
- <https://www.mathworks.com/help/images/single-image-super-resolution-using-deep-learning.html>

