Pyramids

Pablo Mesejo

pmesejo@go.ugr.es

Universidad de Granada

Departamento de Ciencias de la Computación e Inteligencia Artificial

UNIVERSIDAD DE GRANADA

DaSCI

Instituto Andaluz de Investigación en **Data Science** and **Computational Intelligence**

Thinking in Frequency

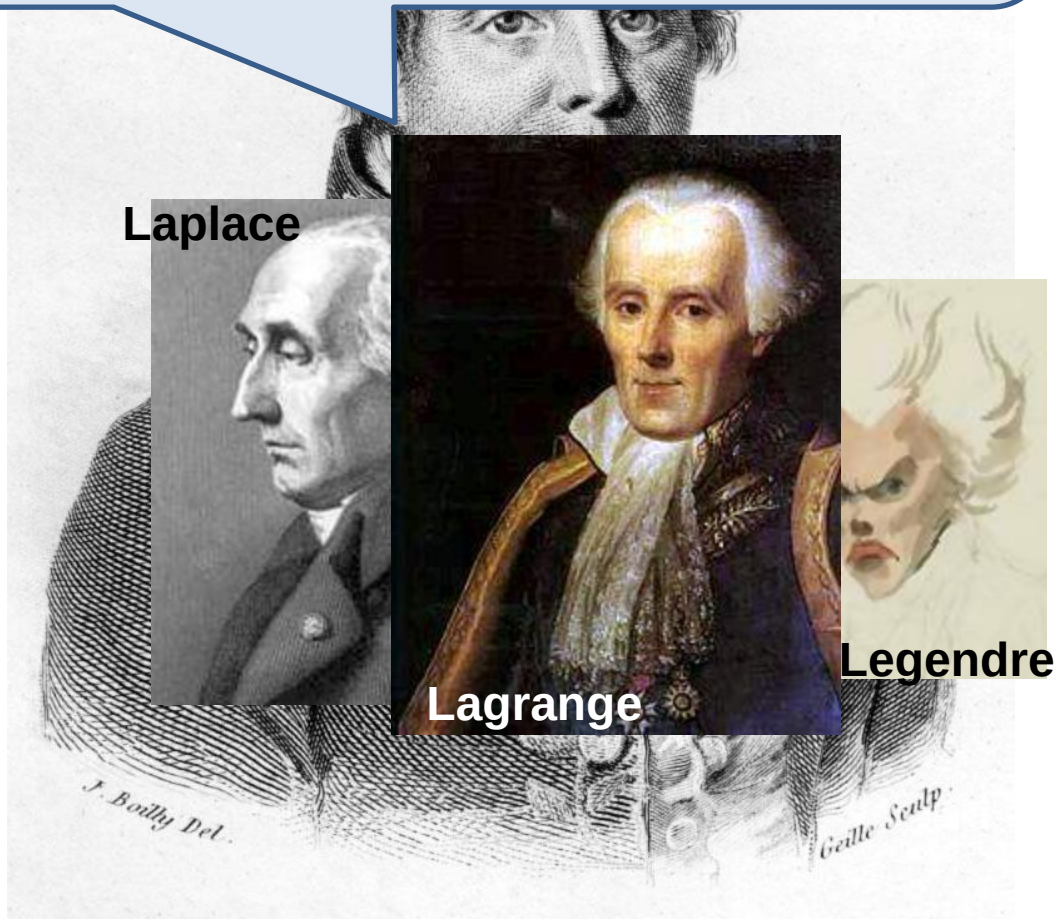
Jean Baptiste Joseph Fourier (1768-1830)

had crazy idea (1807):

Any 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 others
  - Not translated into English until 1878!
- Now, this is called Fourier Series.

...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.



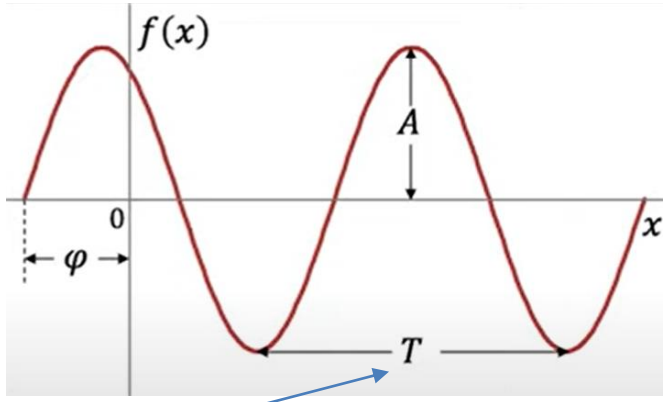
Slide: Efros

A sum of sinusoids

Our building block:

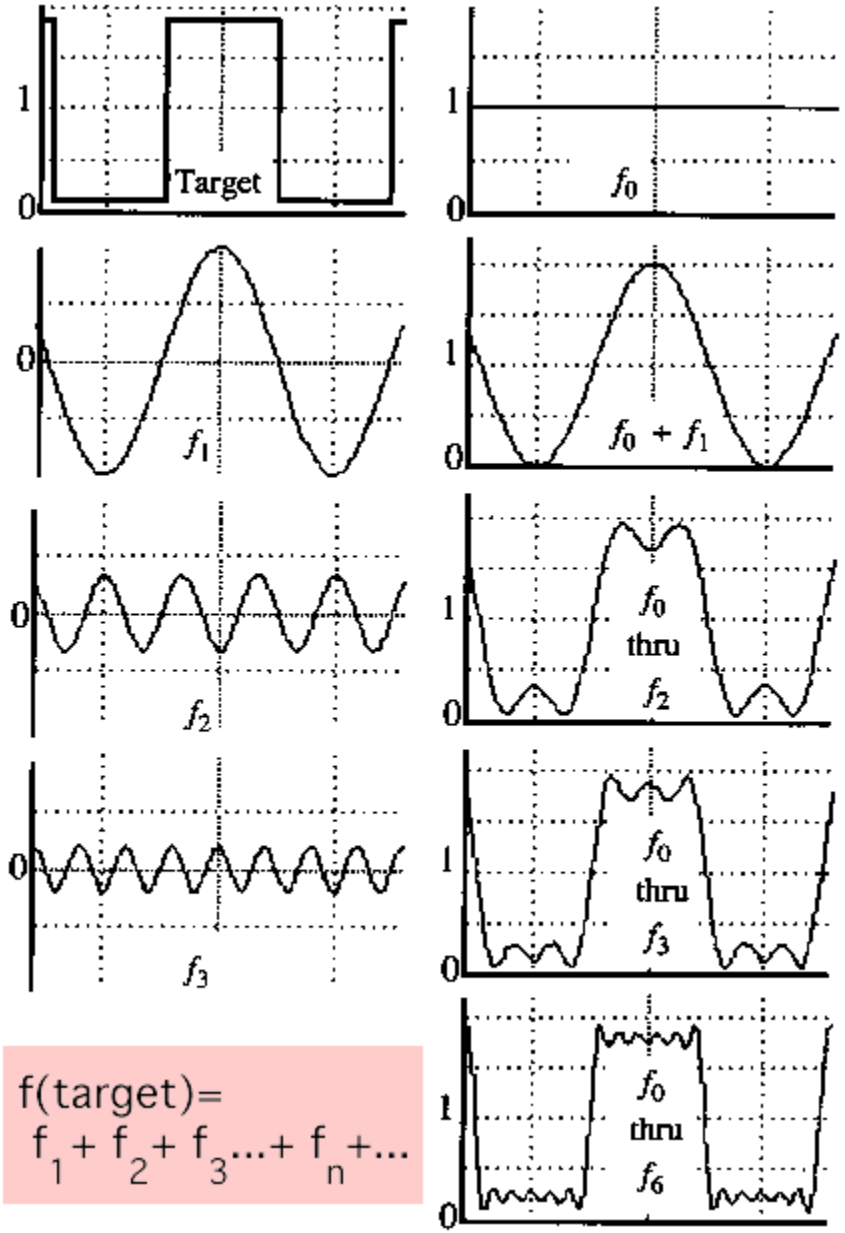
$$f(x) = Asin(2\pi ux + \varphi)$$

Amplitude — Frequency (1/T) — Phase



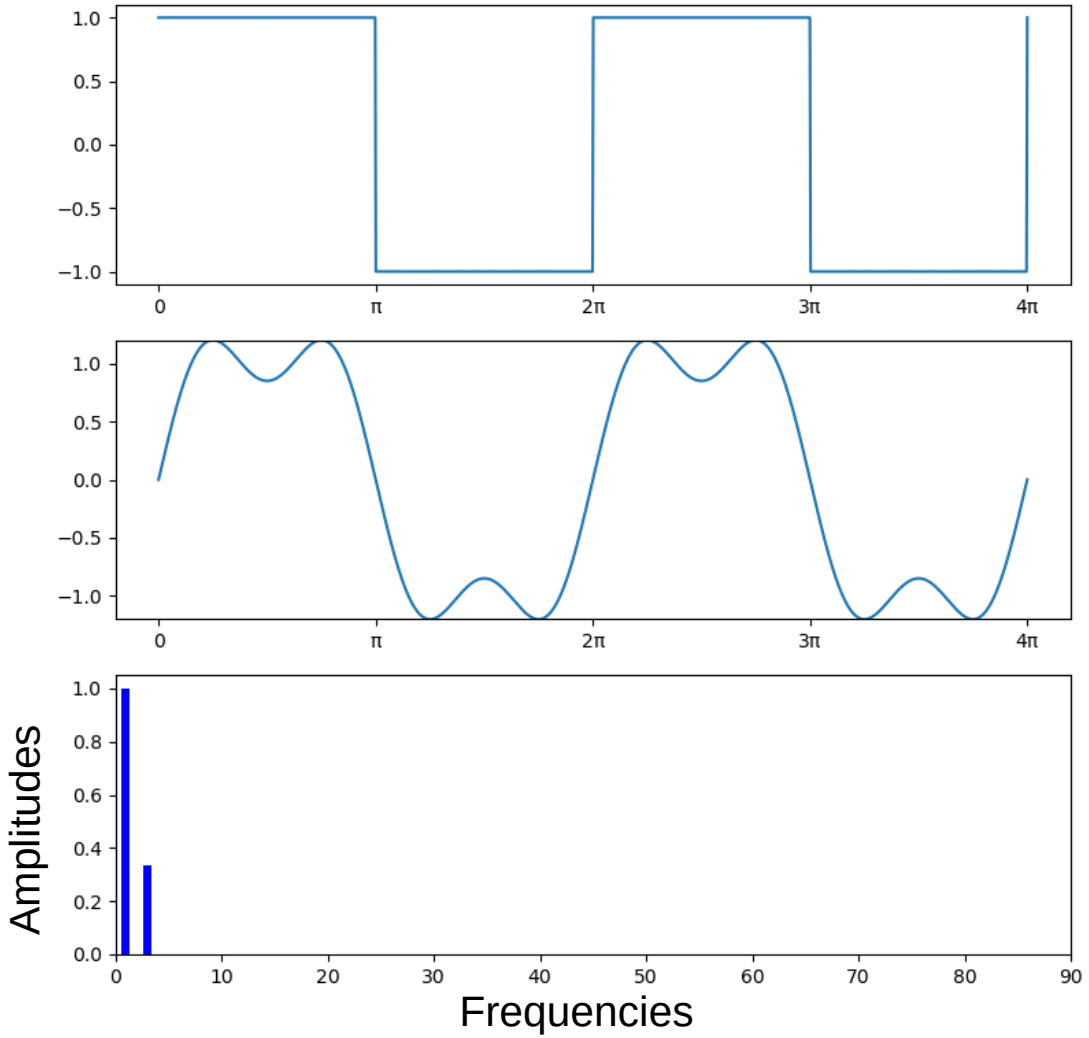
Period

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

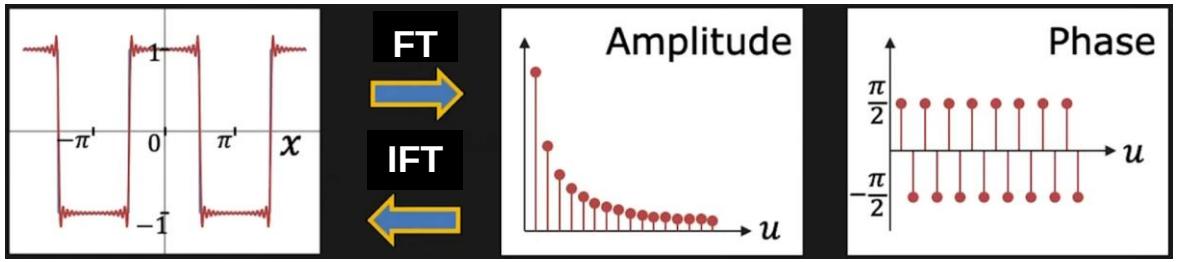


f(target)=
$f_1 + f_2 + f_3 \ldots + f_n + \ldots$

A sum of sinusoids



Frequency representation of signal



The **Fourier Transform (FT)** represents a signal $f(x)$ in terms of Amplitudes and Phases of its constituent sinusoids.

The **Inverse Fourier Transform (IFT)** computes the signal $f(x)$ from the Amplitudes and Phases of its constituent sinusoids.

No loss of information!!!

Slide: Derek Hoiem, Shree Nayar

Fourier Transform is Complex

$f(x)$ → **Fourier Transform** → $F(u)$

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{j2\pi ux} du \qquad F(u) = \int_{-\infty}^{\infty} f(x) e^{-j2\pi ux} dx$$

For every frequency u, $F(u)$ holds magnitude (A) and phase (φ) of the corresponding sinusoid.

How can we do it? For mathematical convenience, we use **complex numbers**!

The **magnitude** tells "how much" of a certain frequency component is present.

The **phase** tells "where" the frequency component is in the signal/image.

Note: For each frequency, the **magnitude** (absolute value) of the complex value represents the **amplitude** of a constituent complex sinusoid with that frequency. We could say that amplitude is the peak value of a sinusoid in the **time/spatial domain**, while magnitude is the absolute value of a given frequency in the **Fourier domain**.

Rectangular or Cartesian form

$$F(u) = R(u) + jI(u)$$

$$A = |F(u)| = \sqrt{R(u)^2 + I(u)^2} \qquad \varphi = tan^{-1}\frac{I(u)}{R(u)}$$

⟺

Polar form

$$F(u) = Ae^{j\varphi}$$
$$= A(cos\varphi + jsin\varphi)$$

Complex Exponential (Euler's Formula)

Convolution Theorem

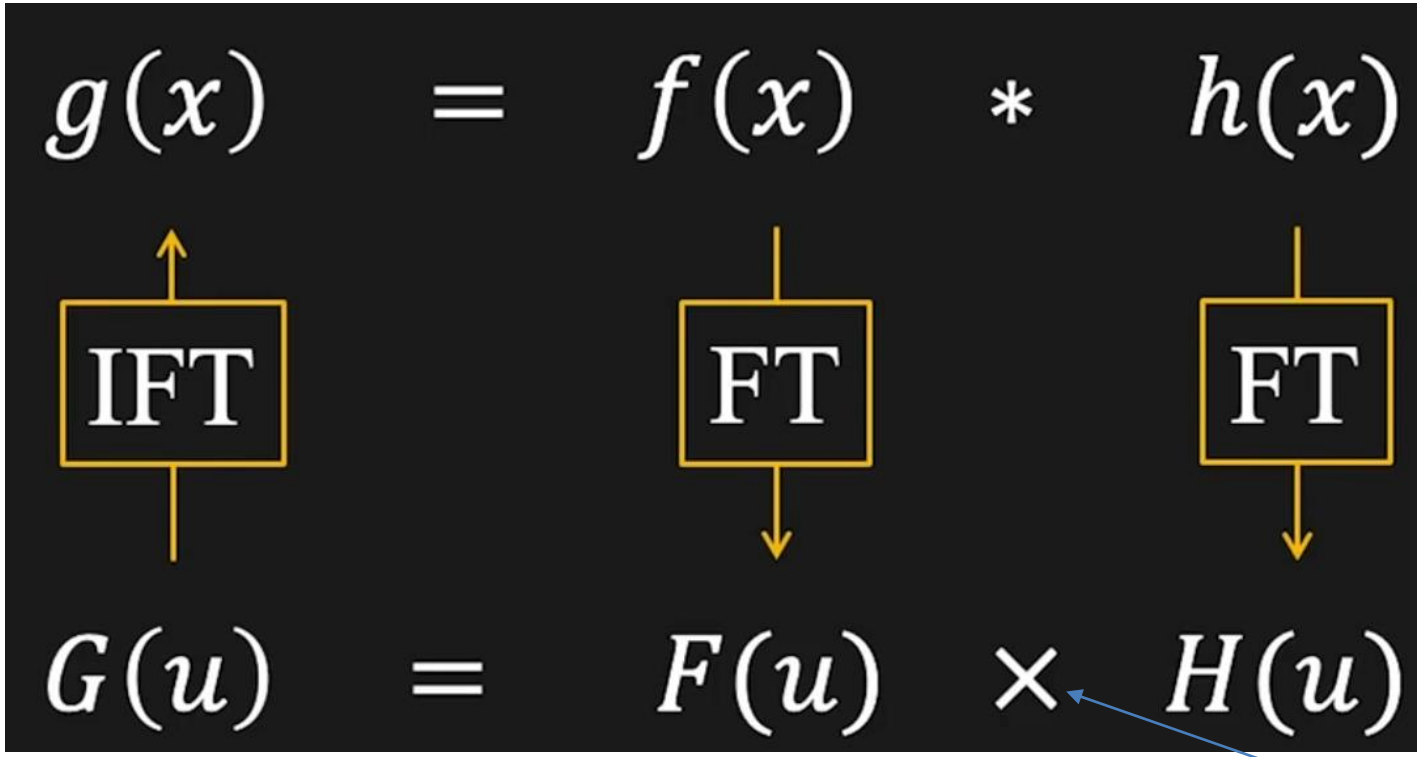
- The Fourier Transform of the convolution of two functions is equal to the pointwise multiplication of their Fourier Transforms.

$$FT(g * h) = FT(g)FT(h)$$

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

Remember: correlation does not allow a clear interpretation in terms of frequency!

Convolution using Fourier Transform

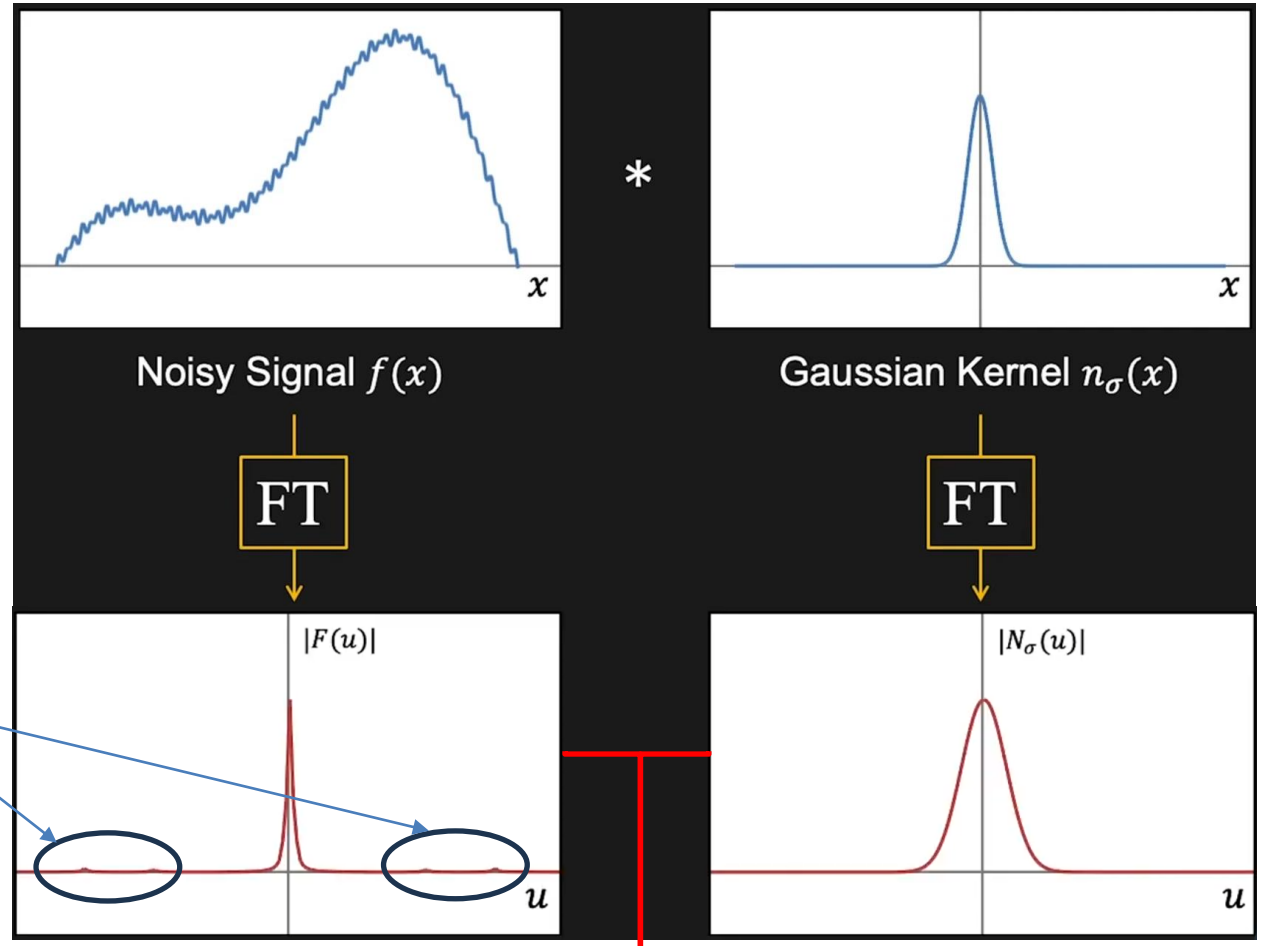


Why is all this useful?

1) It's generally faster (particularly for large filters)! See Heckbert and Kundur notes regarding *Fast Fourier Transform*. In 1D convolution (FFT vs direct): $O(NlogN)$ vs $O(N^2)$.
2) This perspective allows us to interpret image filtering in terms of frequencies!

Slide: Shree Nayar

Interpreting filtering in terms of frequencies



High-frequency **noise** we'd like to remove

Low-pass filter: it "kills" all high frequencies (they are multiplied by zero)

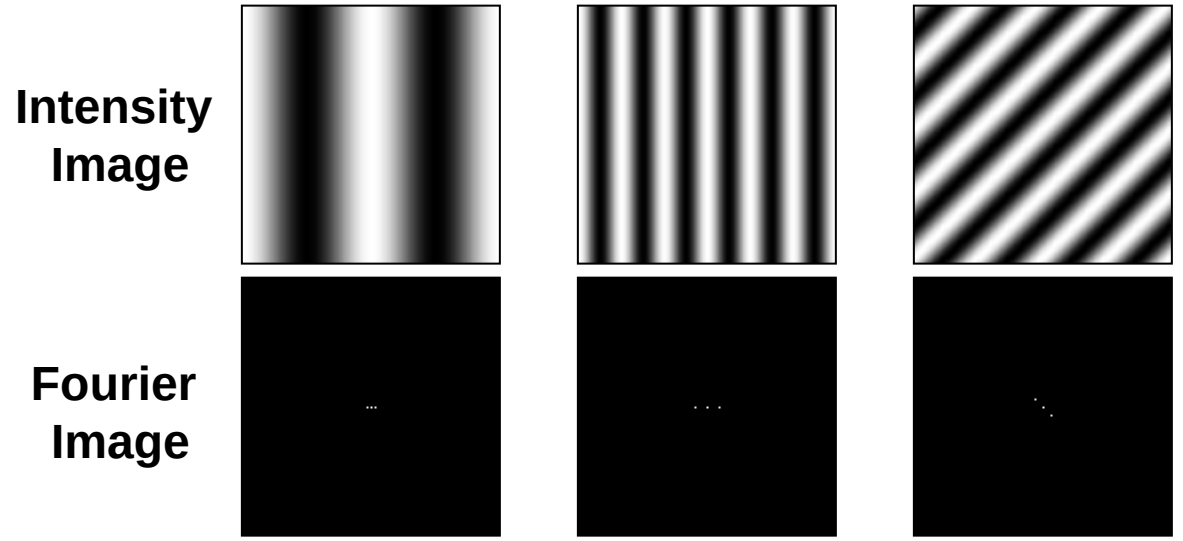
Slide: Shree Nayar

Fourier Analysis in Images

- According to Fourier, signals can be decomposed into sums of sines and cosines. But... can this be applied to images? 🤔
- Of course! An image is a visual signal!

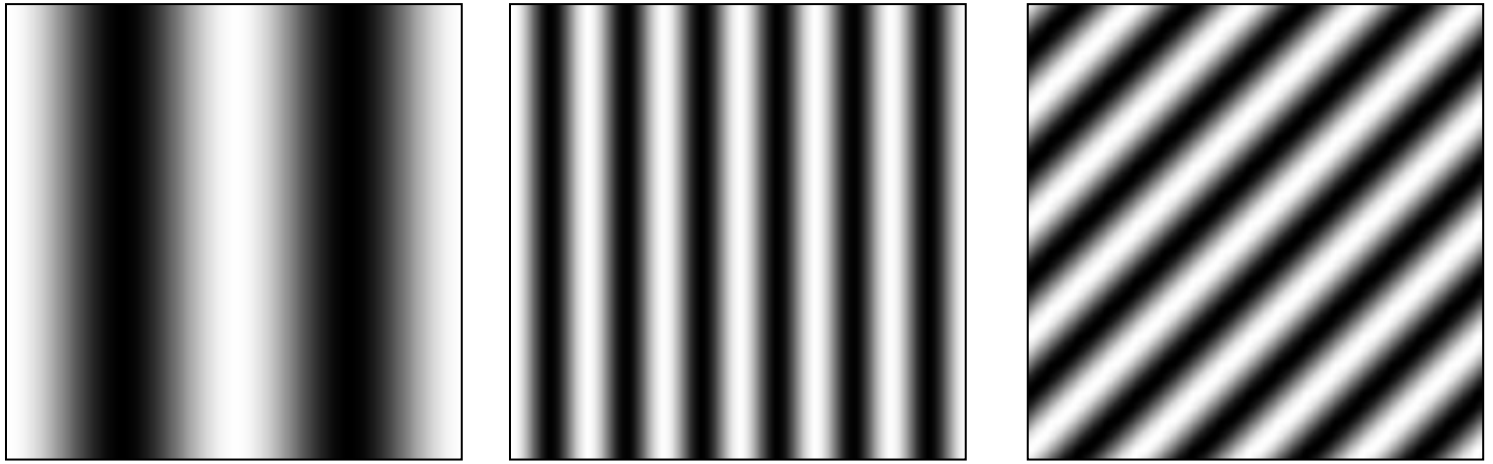
Any signal can be expressed as a sum of a series of sinusoids

Note: Fourier Images are just magnitude images (we generally do not display phase images)



In our case, these are sinusoidal variations in brightness across the image (i.e. image information is represented not for each pixel separately but rather for each frequency)

Fourier Analysis in Images

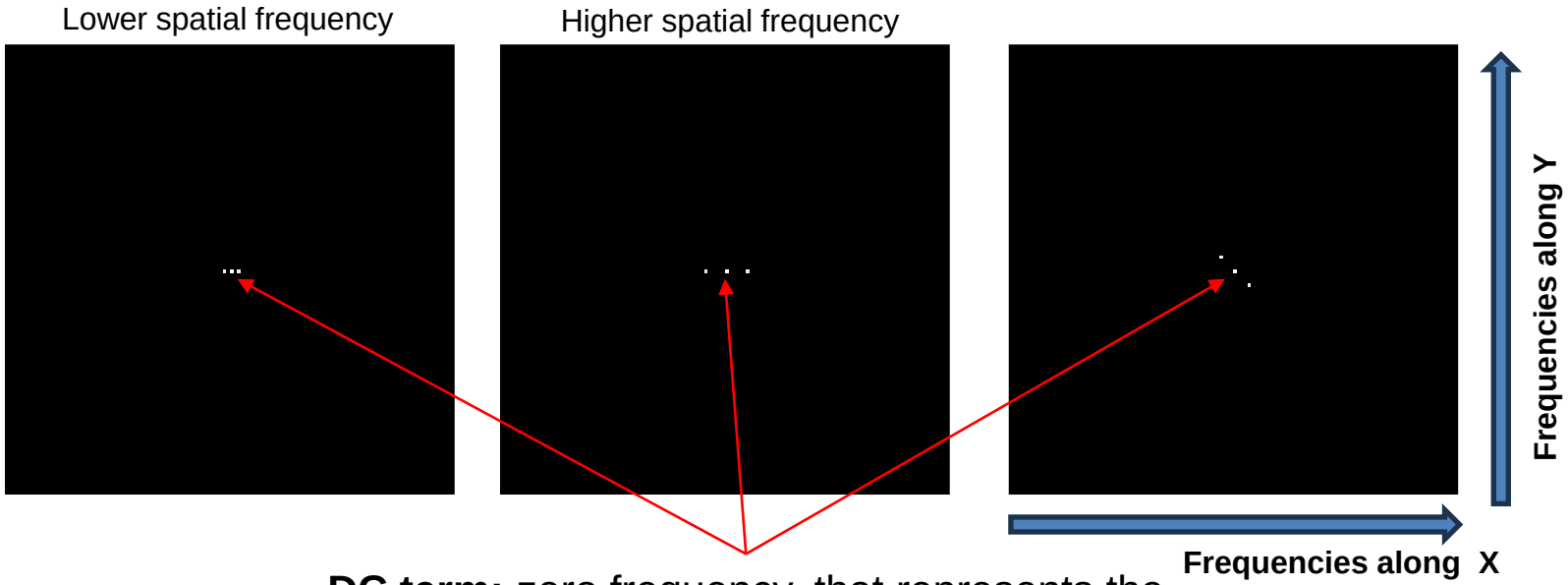


Intensity Image

Two bright pixels, either side of the center, encode the (single) sinusoidal pattern (i.e. frequencies +*k* and –*k*).

Every pixel of the Fourier image is a spatial frequency value. The magnitude of that value (i.e. amplitude/contrast of a constituent sinusoid) **is encoded by the brightness of the pixel.**

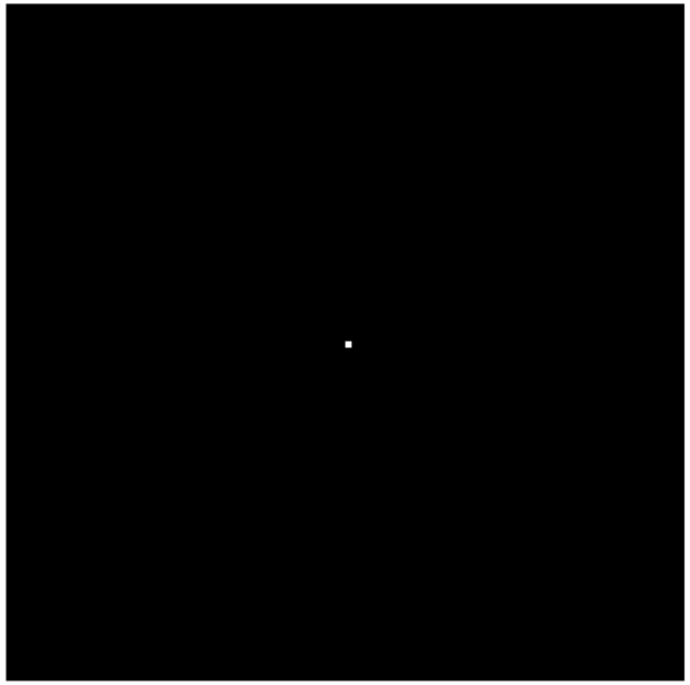
Since there is only one Fourier component in these images, all other values in the Fourier image are zero (black)

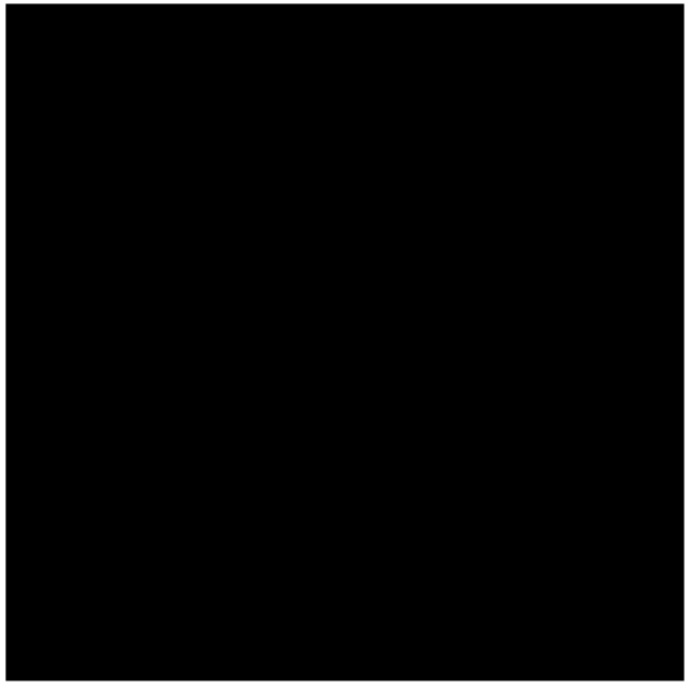


Fourier Image

DC term: zero frequency, that represents the average brightness of the whole image

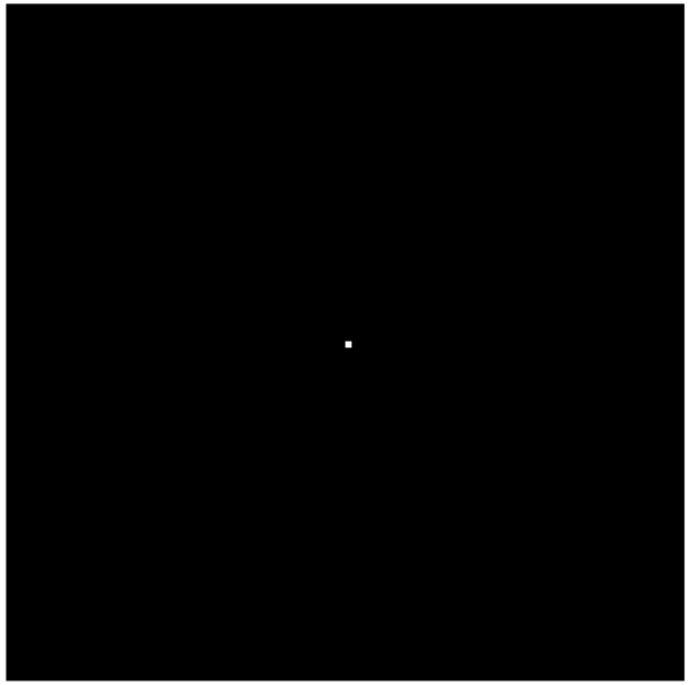
How change in frequency affects signal

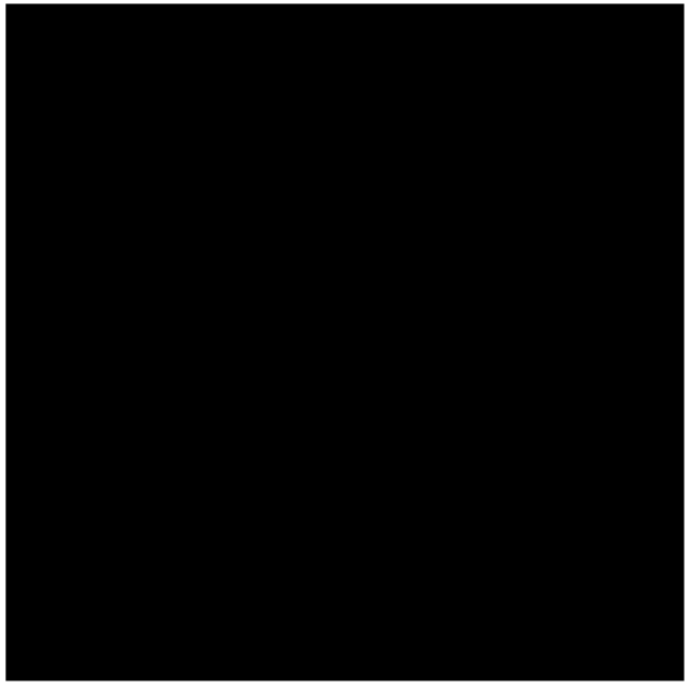




Slide: Derek Hoiem

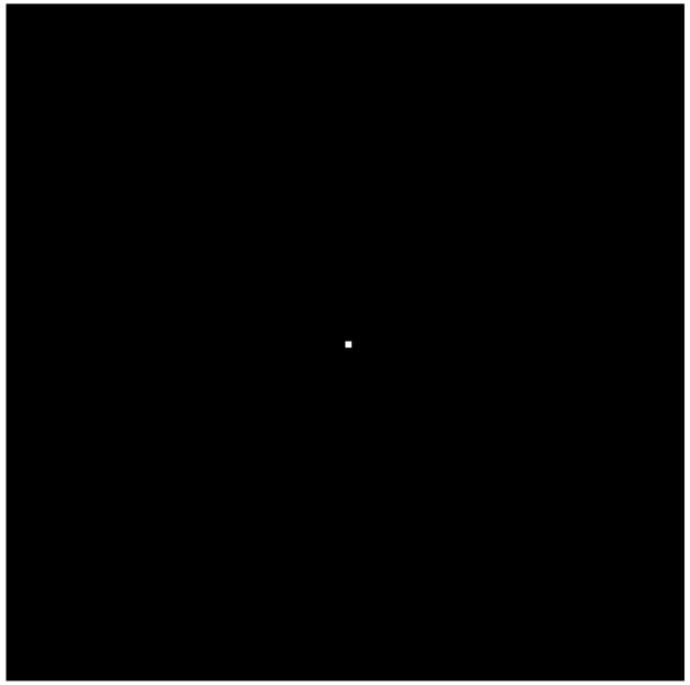
How change in frequency affects signal

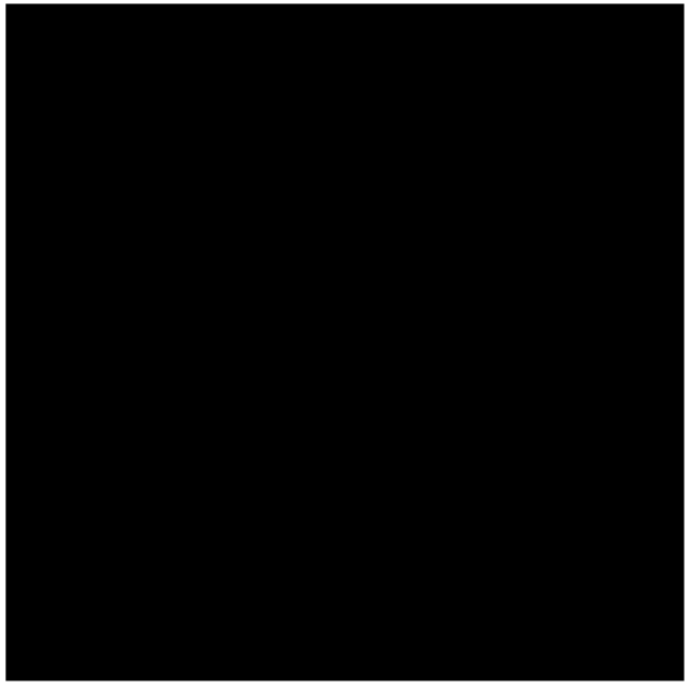




Slide: Derek Hoiem

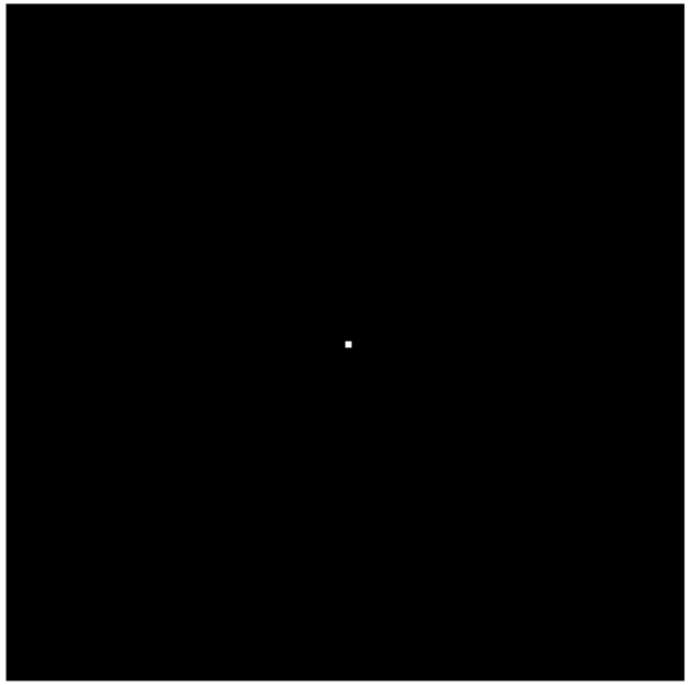
How change in frequency affects signal

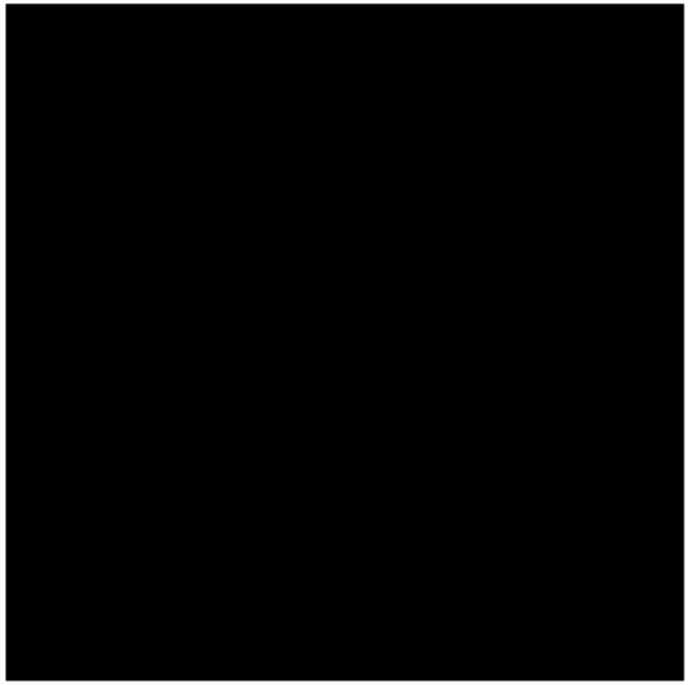




Slide: Derek Hoiem

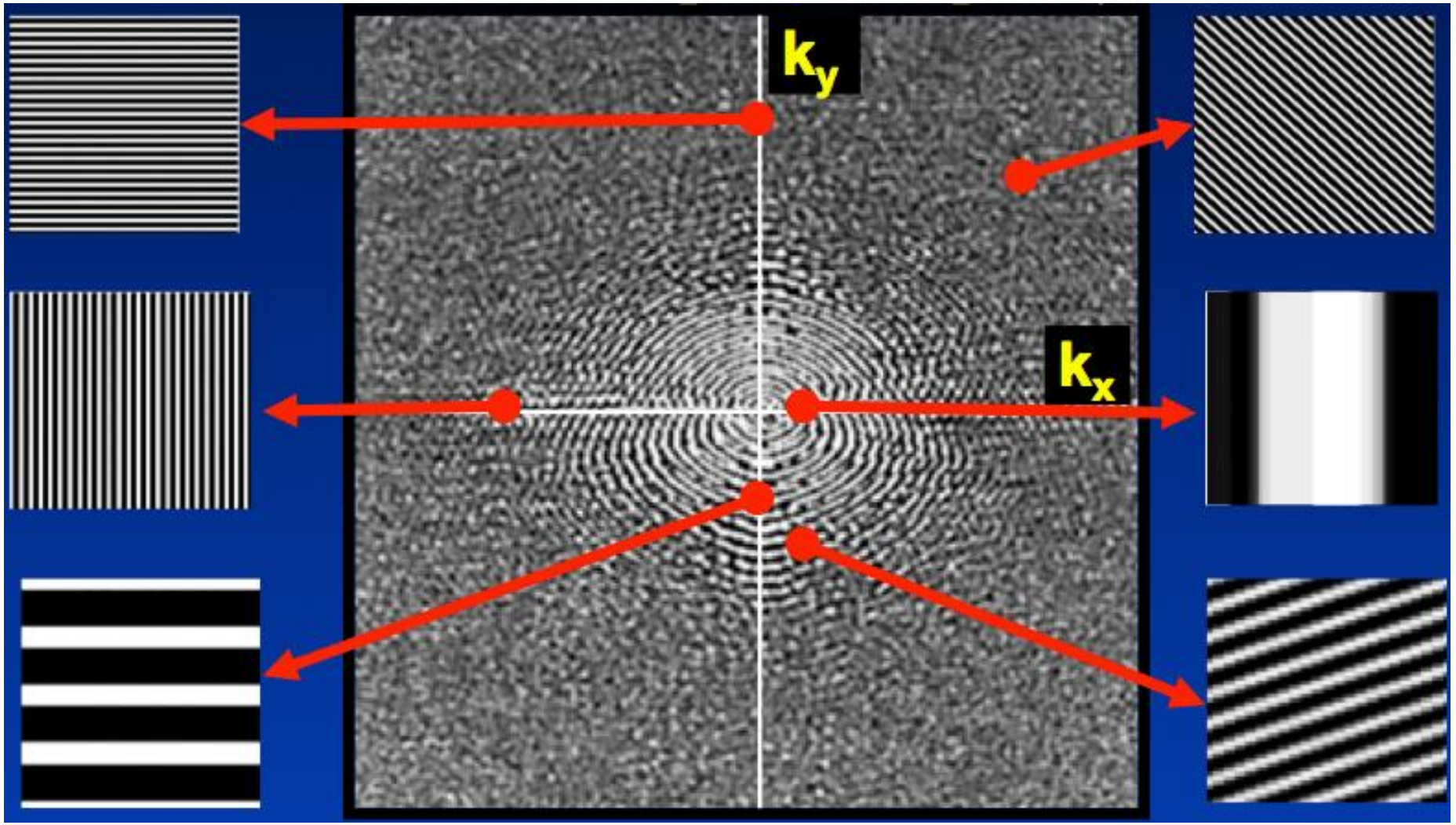
How change in frequency affects signal





Slide: Derek Hoiem

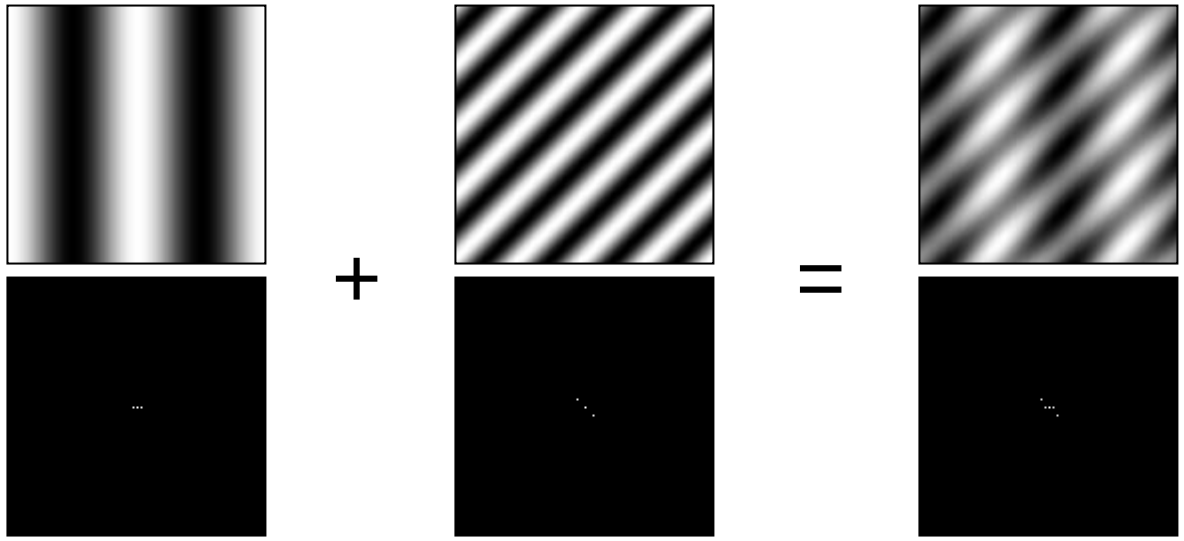
How change in frequency affects signal



Each point in the Fourier Image corresponds to a particular spatial frequency

Slide: Karla Miller

Signals can be composed

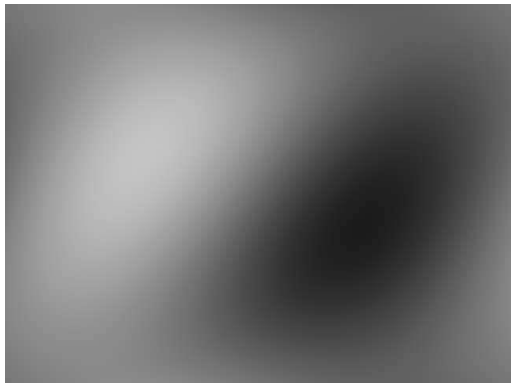


The Fourier representation allows manipulation of the global information content of the image by spatial manipulations of the transformed image.

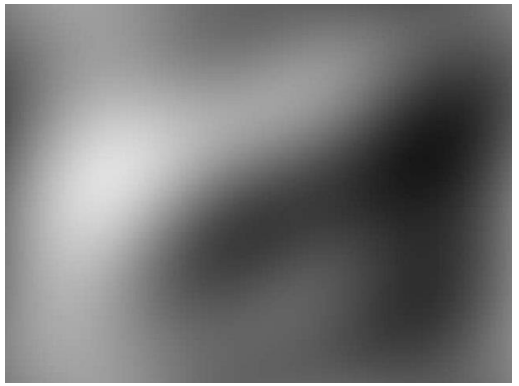
Original

First Frequency (Average)

Second Frequency



Third Frequency



Using 50% of frequencies

https://photo.stackexchange.com/questions/40401/what-does-frequency-mean-in-an-image

Fourier Analysis in Images (technical note)

```
frequency_x = 0                # Frequency in X-axis
frequency_y = 0.25             # Frequency in Y-axis
amplitude = 100.0              # Sinusoid amplitude
phase = np.pi/4                # Sinusoid phase

# We generate coordinates
x = np.linspace(-10, 10, 100)
y = np.linspace(-10, 10, 100)
x, y = np.meshgrid(x, y)

# We generate the 2D sinusoid
sinusoid_2D = amplitude * np.sin(2 * np.pi * (frequency_x * x + frequency_y * y) + phase)

# Compute the 2-dimensional discrete Fourier Transform.
FT = np.fft.fft2(sinusoid_2D) #https://numpy.org/doc/stable/reference/generated/numpy.fft.fft2.html
# Shift the zero-frequency component to the center of the spectrum.
shifted_FT = np.fft.fftshift(FT) #https://numpy.org/doc/stable/reference/generated/numpy.fft.fftshift.html
# Compute magnitude
magnitude = np.abs(shifted_FT) #sometimes, for visualization purposes, it's convenient to do np.log(np.abs(shiftedFT))

# Display
plt.figure(figsize=(10, 5))
plt.subplot(121), plt.imshow(sinusoid_2D, cmap='gray')
plt.title('Sinusoid'), plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(magnitude, cmap='gray')
plt.title('Fourier Image'), plt.xticks([]), plt.yticks([])
plt.show()
```

Fourier Analysis in Images (technical note)

- Filtering with Fast Fourier Transform (FFT)

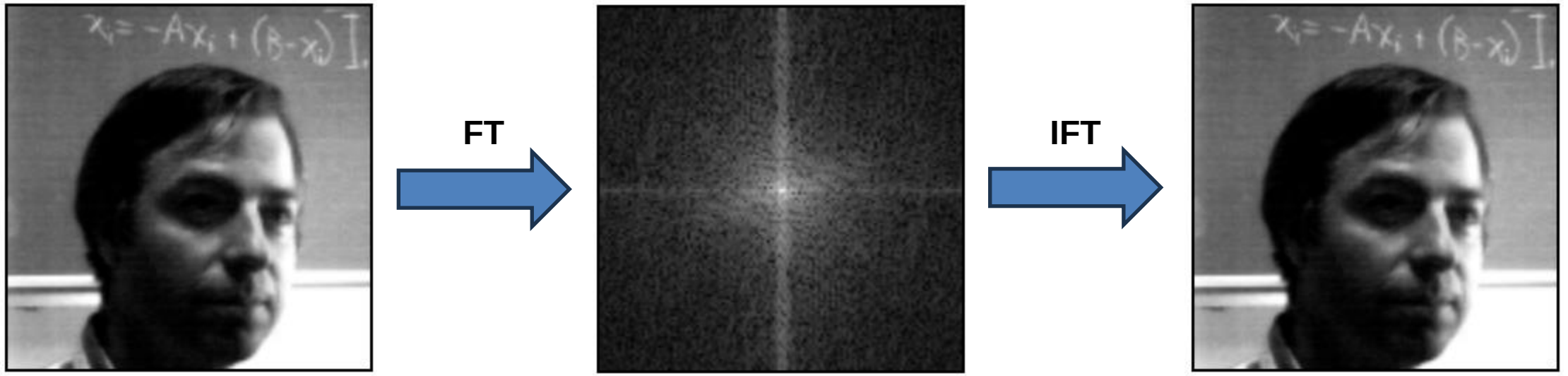
```
import matplotlib.pyplot as plt
import numpy as np

def filter_image(im, fil):
    '''
    im: H x W floating point numpy ndarray representing image in grayscale
    fil: M x M floating point numpy ndarray representing 2D filter
    '''
    H, W = im.shape
    hs = fil.shape[0] // 2 # half of filter size
    fftsize = 2048         # should be order of 2 (for speed) and include padding
    im_fft = np.fft.fft2(im, (fftsize, fftsize))   # 1) fft im with padding
    fil_fft = np.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 = np.fft.ifft2(im_fil_fft)              # 4) inverse fft2
    im_fil = im_fil[hs:hs + H, hs:hs + W]          # 5) remove padding
    im_fil = np.real(im_fil)                       # 6) extract out real part
    return im_fil
```

Slide: Derek Hoiem

Fourier Analysis in Images

An inverse Fourier transform of the Fourier image produces an exact pixel-for-pixel replica of the original brightness image.

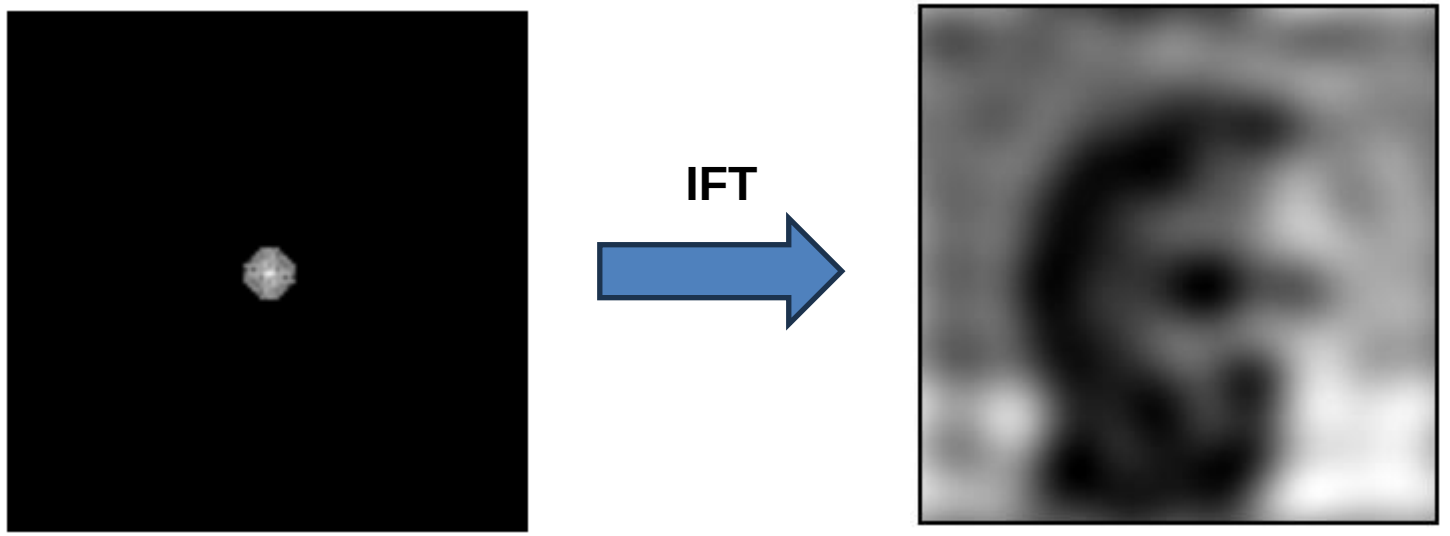


https://web.archive.org/web/20130513181427id_/http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering

Fourier Analysis in Images

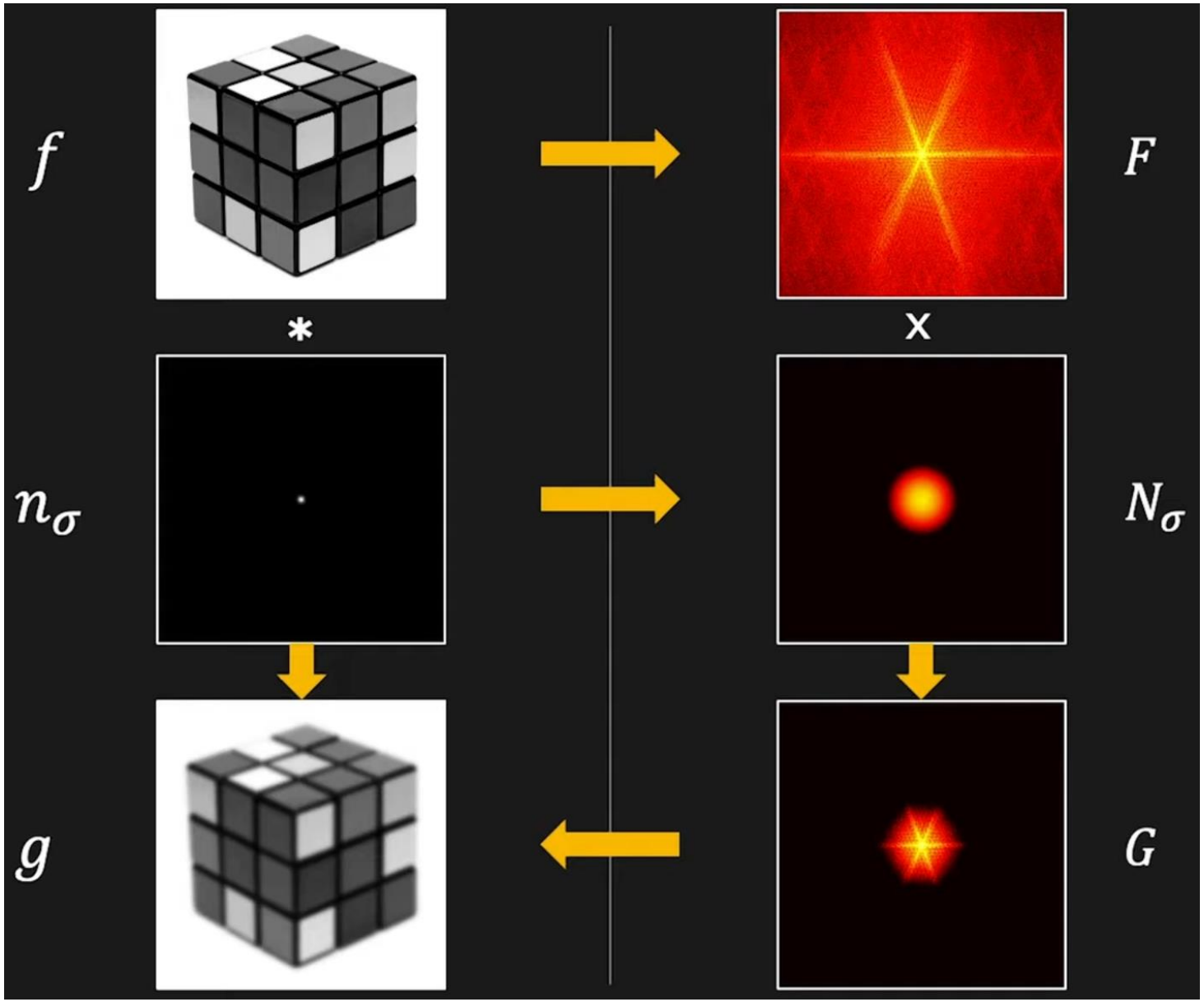
Low frequency components are found near the central DC point → we have to define a radius around the DC point, and zero-out everything else.

The low-pass filtered transform is identical to the central portion of the Fourier transform.



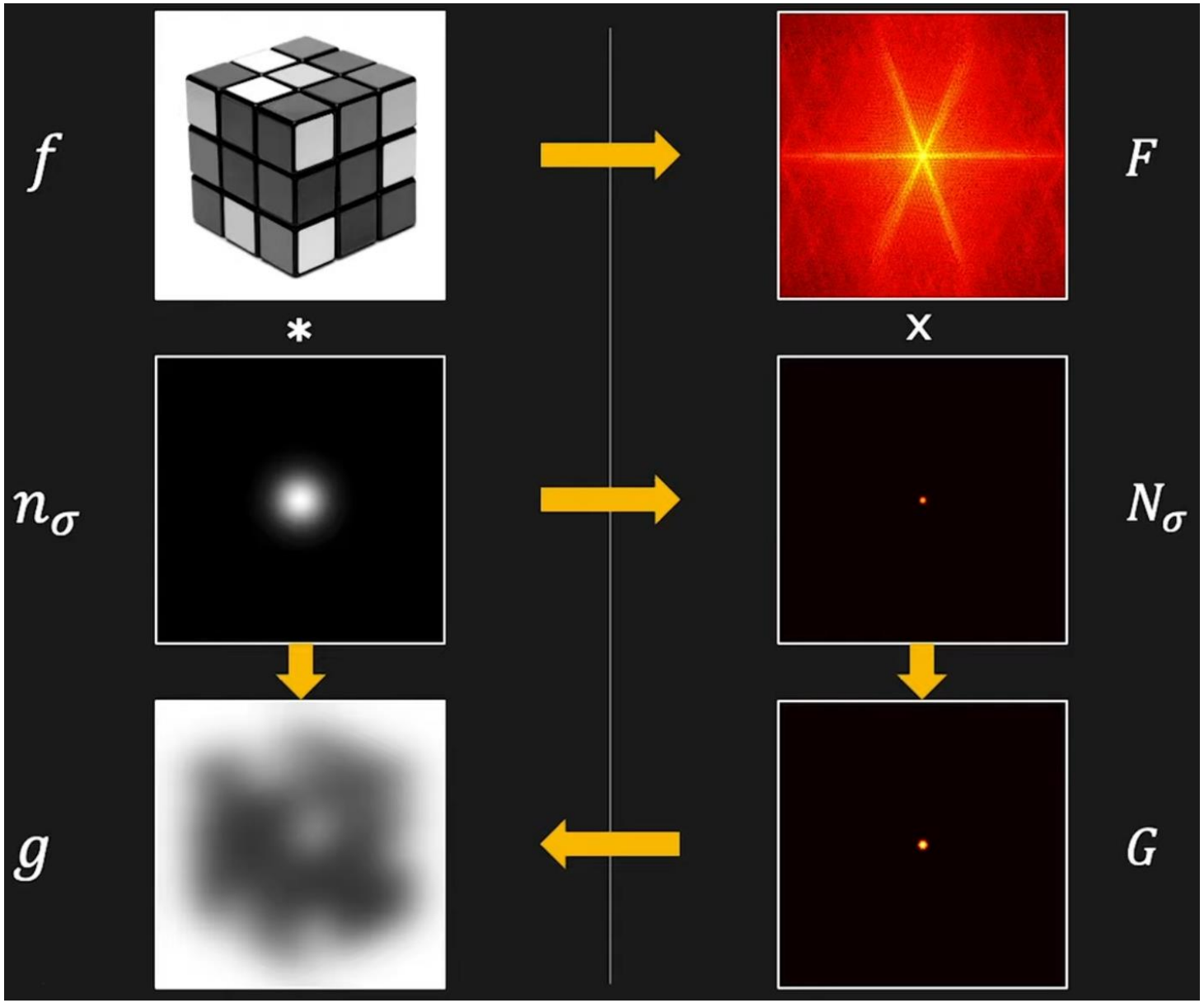
https://web.archive.org/web/20130513181427id_/http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering

Fourier Analysis in Images



Slide:
Shree Nayar

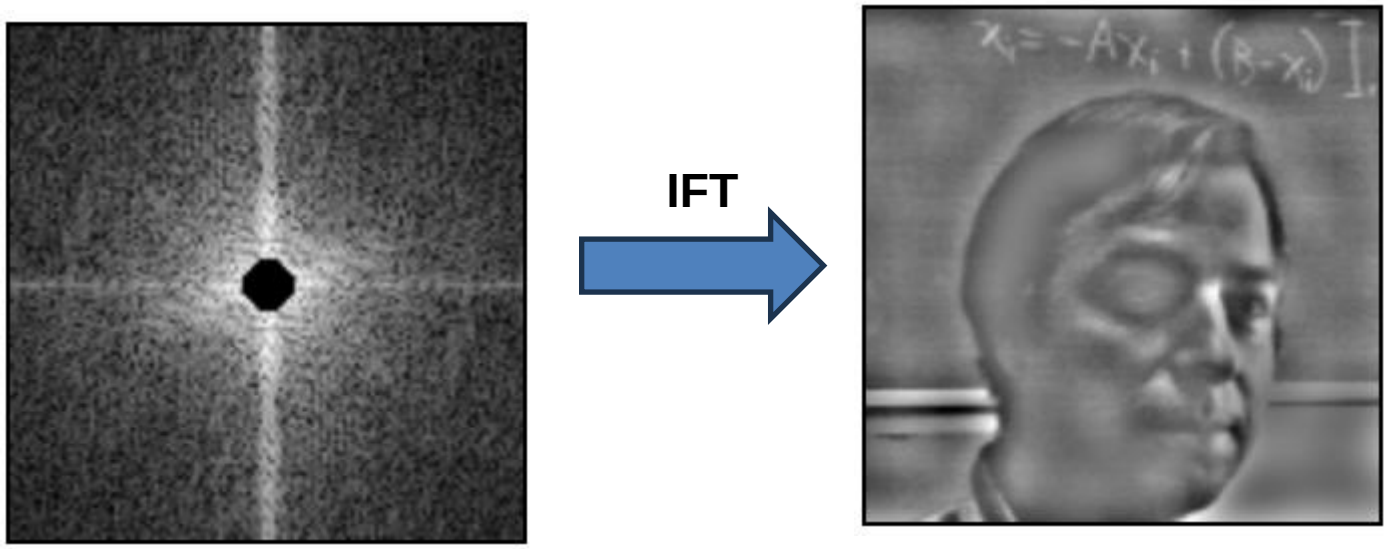
Fourier Analysis in Images



Slide:
Shree Nayar

Fourier Analysis in Images

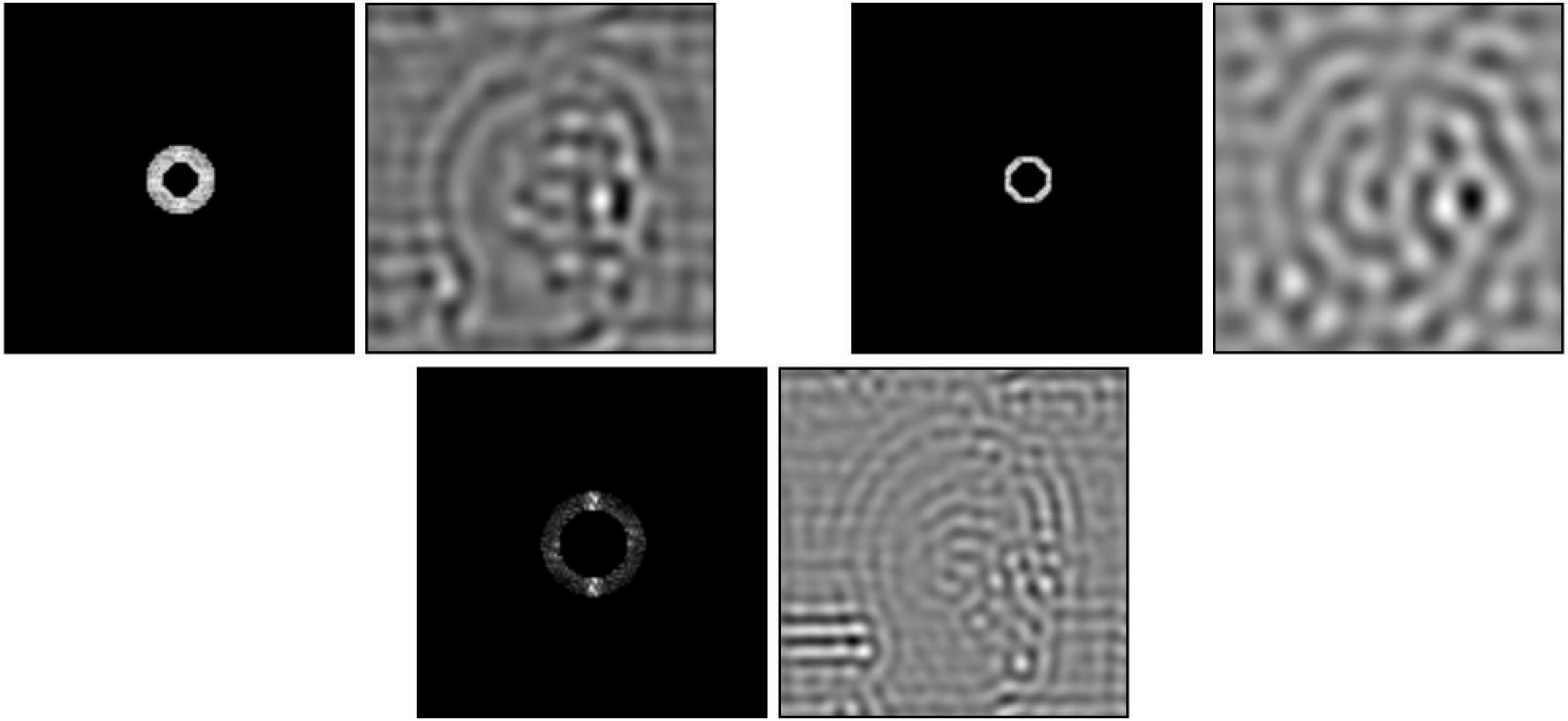
Same idea for high-pass filtering. All spatial frequency components that fall within that radius are eliminated, preserving only the higher spatial frequency components



https://web.archive.org/web/20130513181427id_/http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering

Fourier Analysis in Images

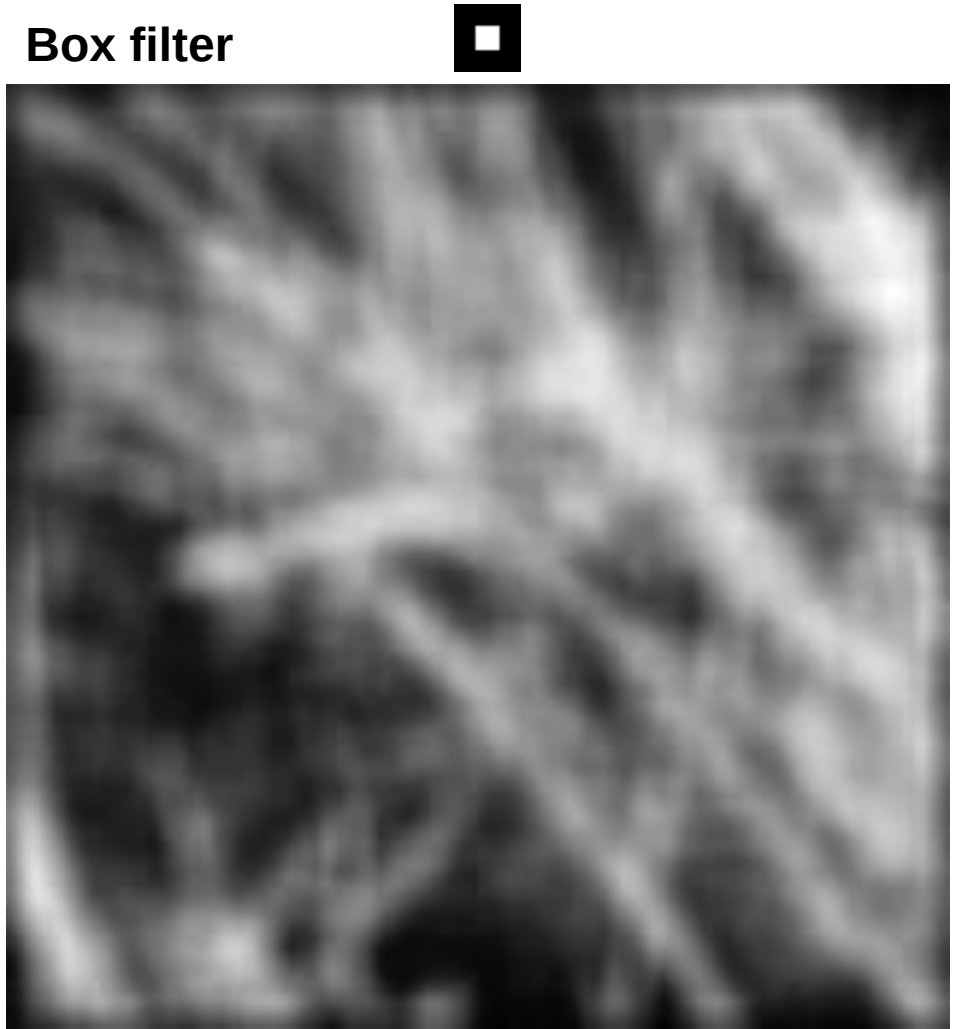
Same for band-pass filters…

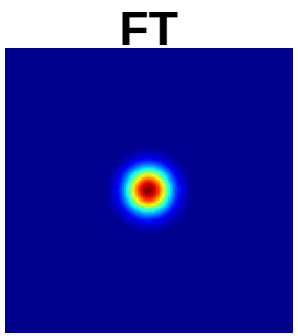


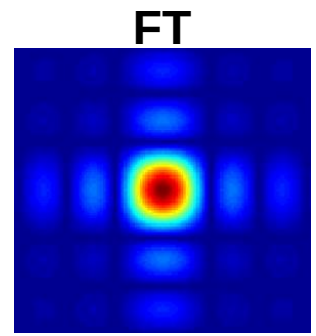
https://web.archive.org/web/20130513181427id_/http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering

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

Let's try to think a bit in frequency terms!



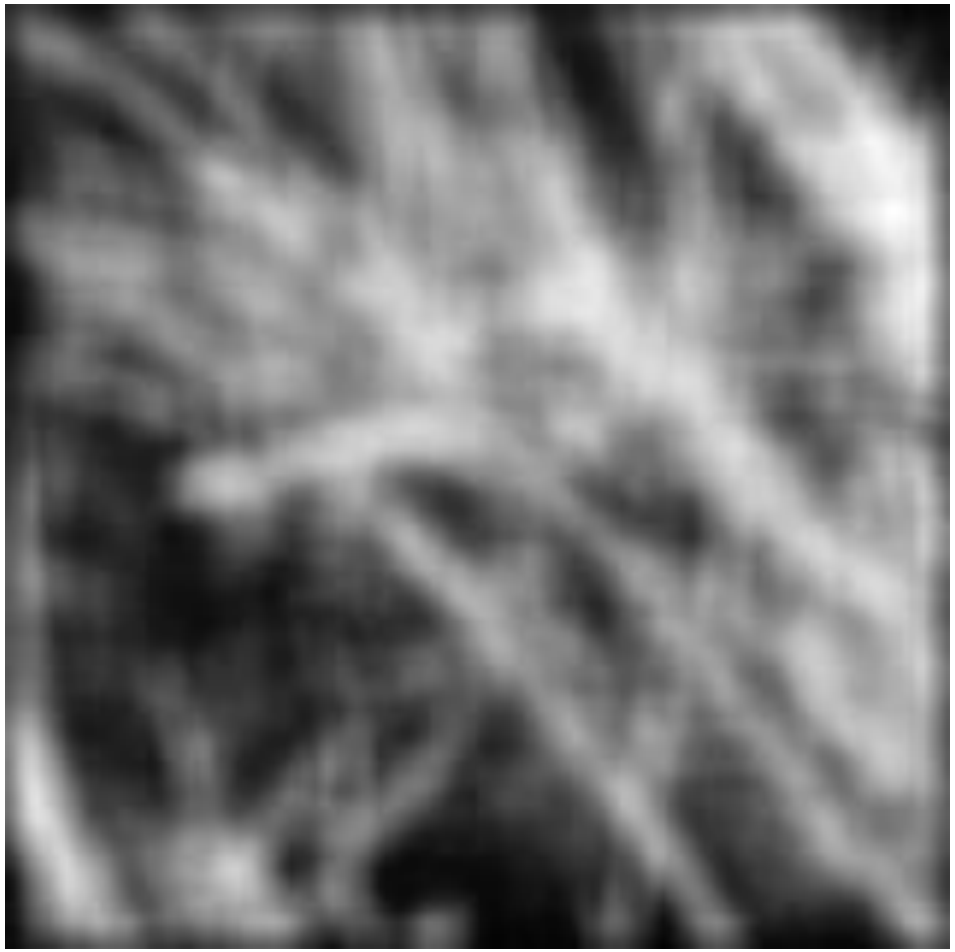




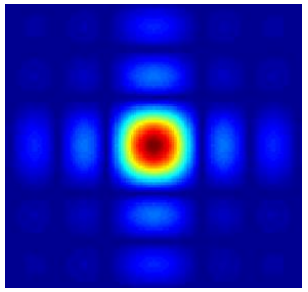
Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?

Let's try to think a bit in frequency terms!

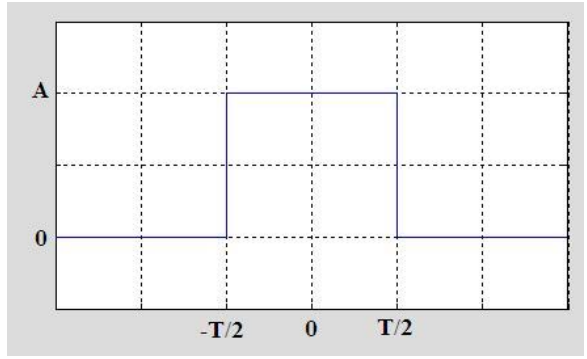
Box filter



FT

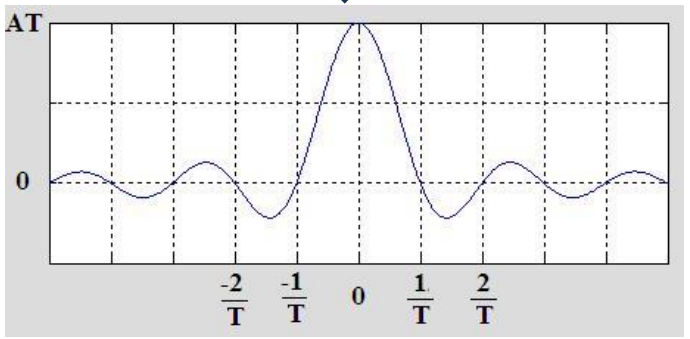


This makes a lot of sense when you consider that the FT of a box function (square pulse) is the sinc function!



FT

$$\operatorname{sinc} x = \frac{\sin(\pi x)}{\pi x}.$$

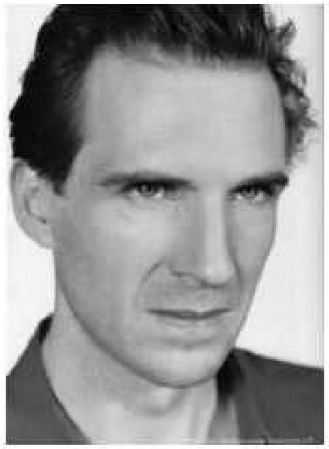


But… what about the phase???

"We generally do not display PHASE images because most people who see them shortly thereafter succomb to hallucinogenics or end up in a Tibetan monastery."

"Introduction to Fourier Transforms for Image Processing"
(by John M. Brayer, University of New Mexico):
https://www.cs.unm.edu/~brayer/vision/fourier.html

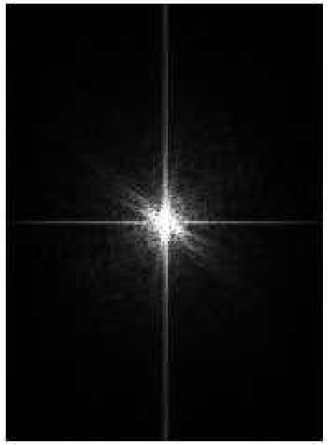
But… what about the phase???

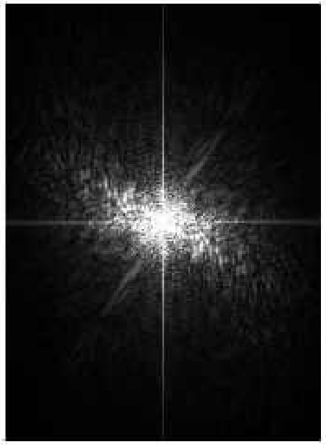


The **magnitude** tells "how much" of a certain frequency component is present.

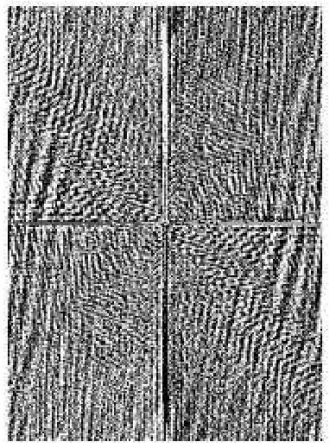
The **phase** tells "where" the frequency component is in the signal/image.

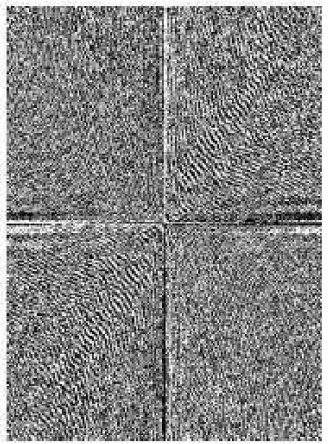
Magnitude





Phase



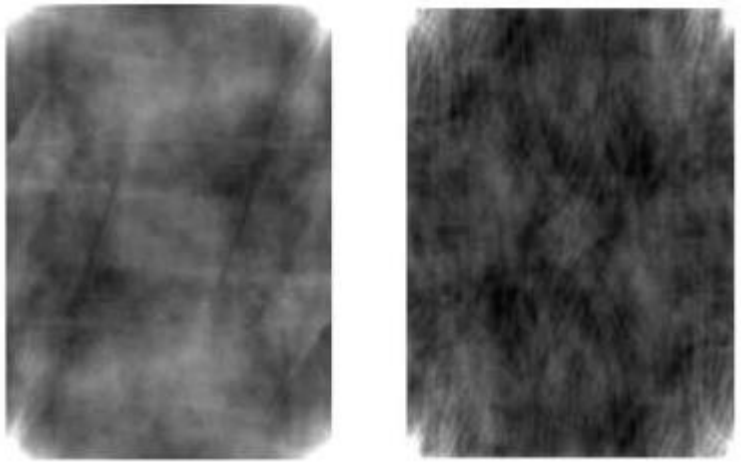


Slide: Deepa Kundur

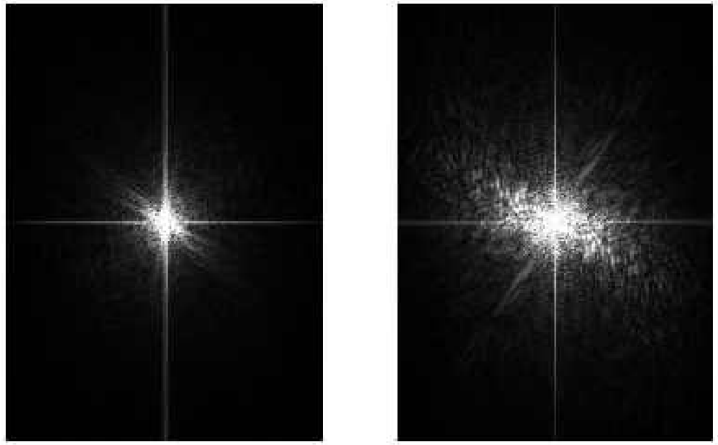
But… what about the phase???

Reconstruction using magnitude only!!!!

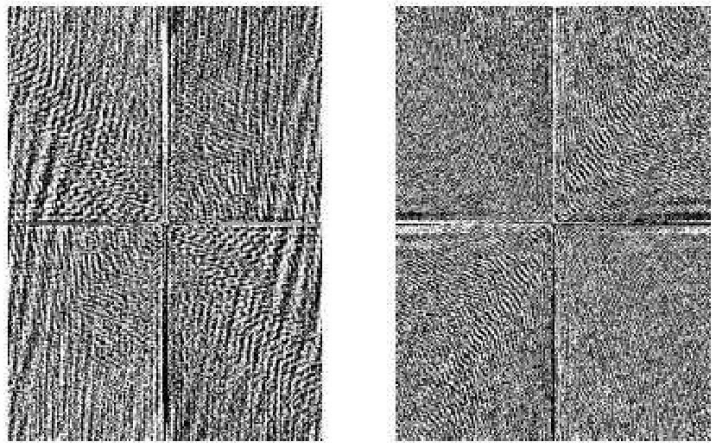
Magnitude is the same; phase=0



Magnitude

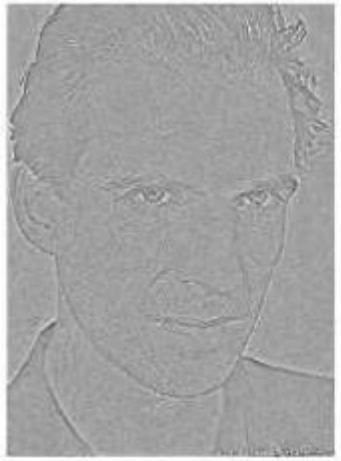


Phase



Slide: Deepa Kundur

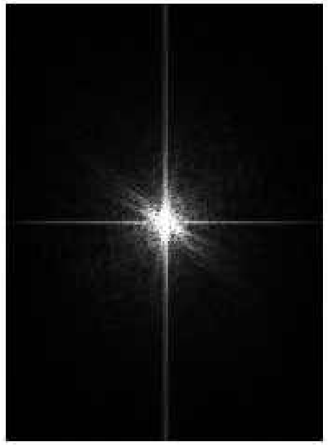
But… what about the phase???

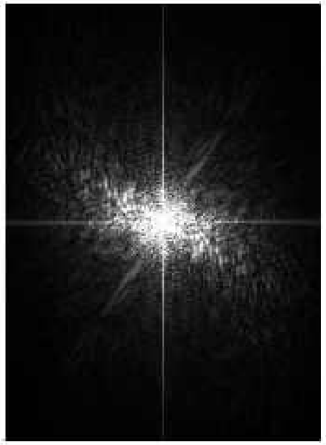


Reconstruction using phase only!!!!

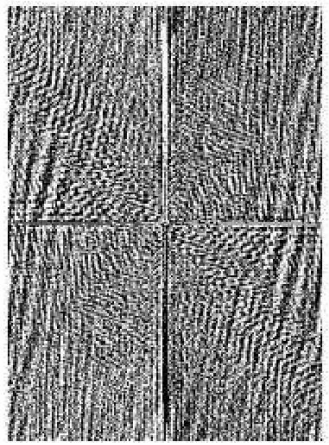
Magnitude normalized to 1;
phase is the same

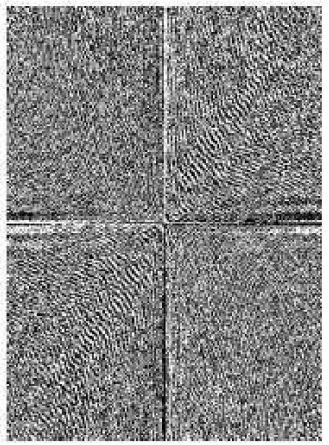
Magnitude





Phase





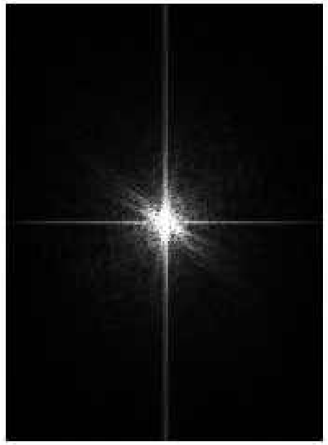
Slide: Deepa Kundur

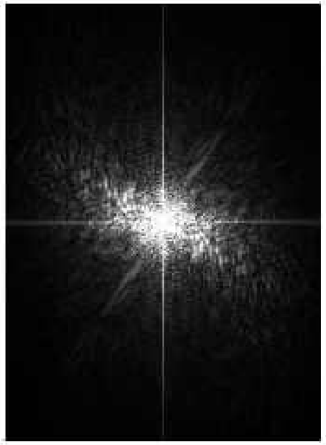
But… what about the phase???

Ralph's phase + Meg's magnitude

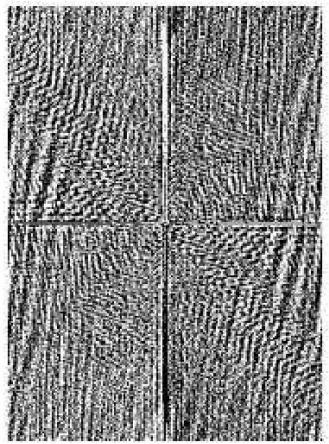
Meg's phase + Ralph's magnitude

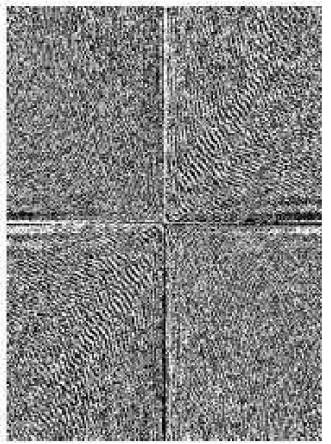
Magnitude





Phase





Slide: Deepa Kundur

But… what about the phase???

Ralph's phase +
Meg's magnitude

Meg's phase +
Ralph's magnitude

Which is more important for a given signal? Magnitude or phase?

Does one component (magnitude or phase) contain more information than another?

🤔

When filtering, if we had to preserve one component (magnitude or phase) more, which one would it be?

Slide: Deepa Kundur

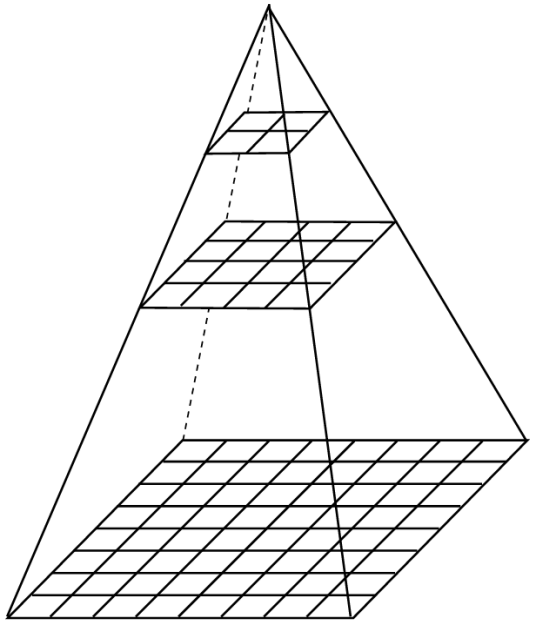
Recommended References

- “An Intuitive Explanation of Fourier Theory” (by Steven Lehar, Boston University): https://web.archive.org/web/20130513181427id_/http://sharp.bu.edu/~slehar/fourier/fourier.html
- “Introduction to Fourier Transforms for Image Processing” (by John M. Brayer, University of New Mexico): https://www.cs.unm.edu/~brayer/vision/fourier.html
- Heckbert, Paul. "Fourier transforms and the fast Fourier transform (FFT) algorithm." Computer Graphics 2.1995 (1995): 15-463. https://www.cs.cmu.edu/afs/andrew/scs/cs/15-463/2001/pub/www/notes/fourier/fourier.pdf
- “Fourier Transform” (by Iain Collings, Macquarie University): https://www.iaincollings.com/signals-and-systems
- “Digital Signal Processing” (by Deepa Kundur, University of Toronto): https://ww2.comm.utoronto.ca/~dkundur//course_info/signals/notes/ and https://ww2.comm.utoronto.ca/~dkundur//course_info/real-time-DSP/notes/
- “First Principles of Computer Vision” (by Shree Nayar, Columbia University): Fourier Transform

Dealing with Scale: Pyramids

Scale-space

- We cannot process images at a single scale.
  - “No single scale is categorically correct: the physical processes that generate signals such as images act at a variety of scales, and none is intrinsically more interesting or important than another.” (Witkin, 1984)
- The most natural structure to operate on multiple scales: a pyramid
  - **Mixing local features from multiple scales** is the best approach since it mimics the Human Visual System (HVS)
- We’ll come back to these ideas when discussing interest point detection and description (e.g. SIFT).



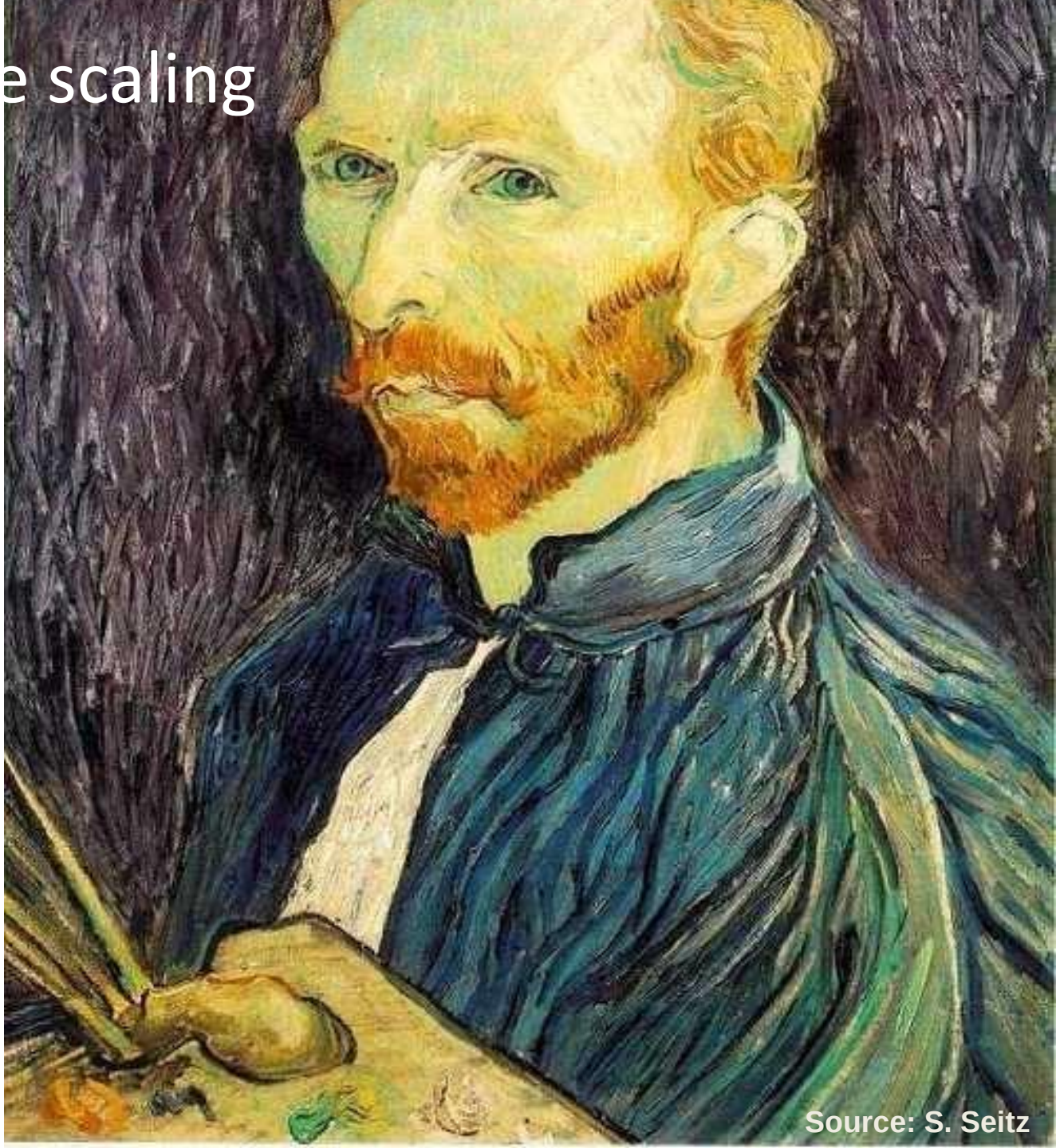
Witkin, A. (1984). *Scale-space filtering: A new approach to multi-scale description*. In *ICASSP'84. IEEE International Conference on Acoustics, Speech, and Signal Processing* (Vol. 9, pp. 150-153).

Lindeberg, T. (1988). *On the construction of a scale-space for discrete images*. KTH Royal Institute of Technology.

Lindeberg, T. (1994). *Scale-space theory: A basic tool for analyzing structures at different scales*. *Journal of applied statistics* 21.1-2 (pp. 225-270).

Image scaling

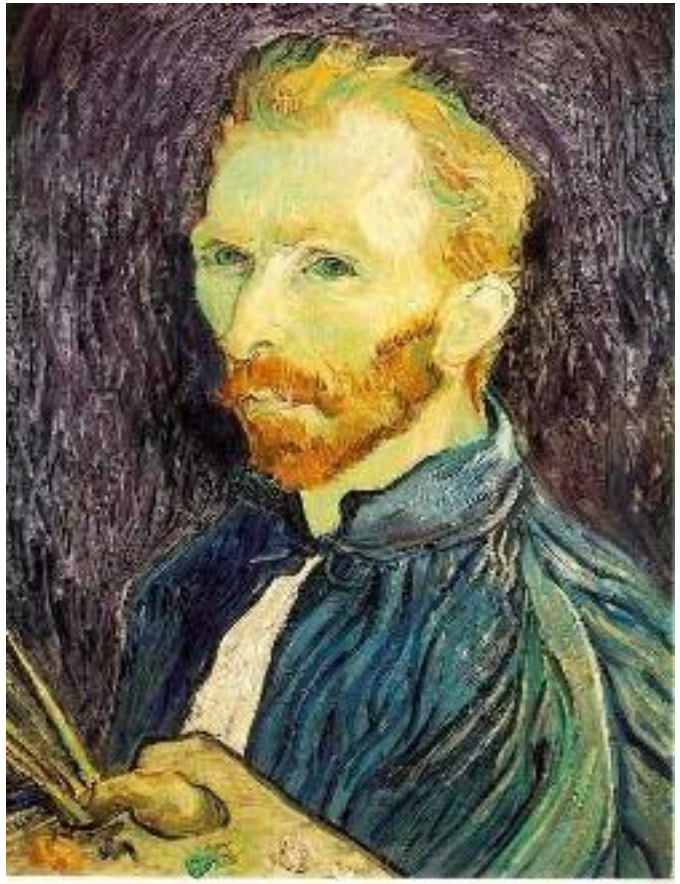
This image is too big to fit on the screen. How can we generate a half-sized version?



Source: S. Seitz

Image sub-sampling

Is this way of operating, regarding the image I obtain, equivalent to taking a photograph at the corresponding distance?



1/4

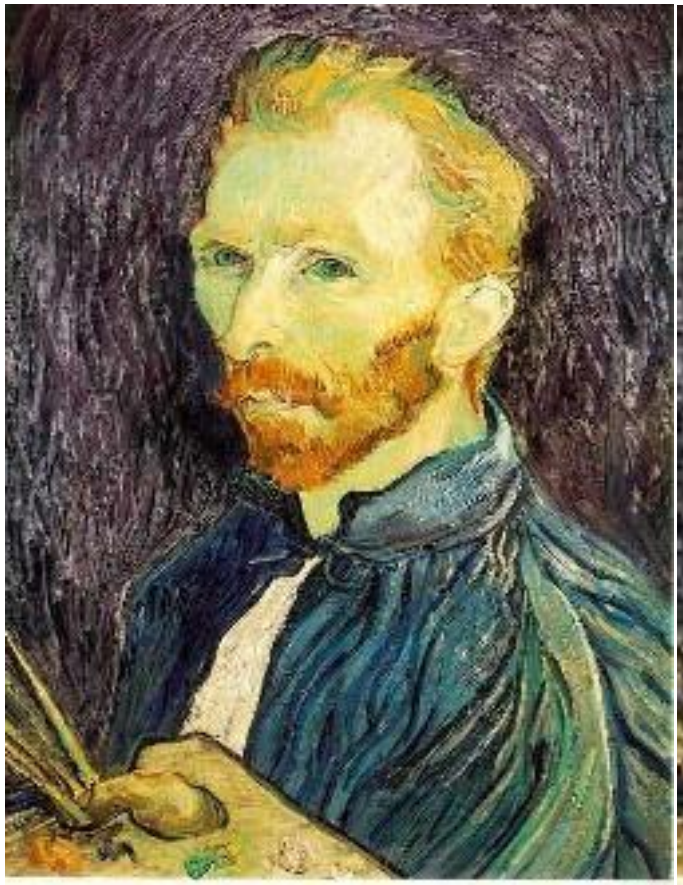
1/8

Throw away every other row and column to create an image with ½ rows and ½ columns

- called ***image sub-sampling***

Source: S. Seitz

Image sub-sampling



1/2

1/4 (2x zoom)

1/8 (4x zoom)

Why does this look so crufty?

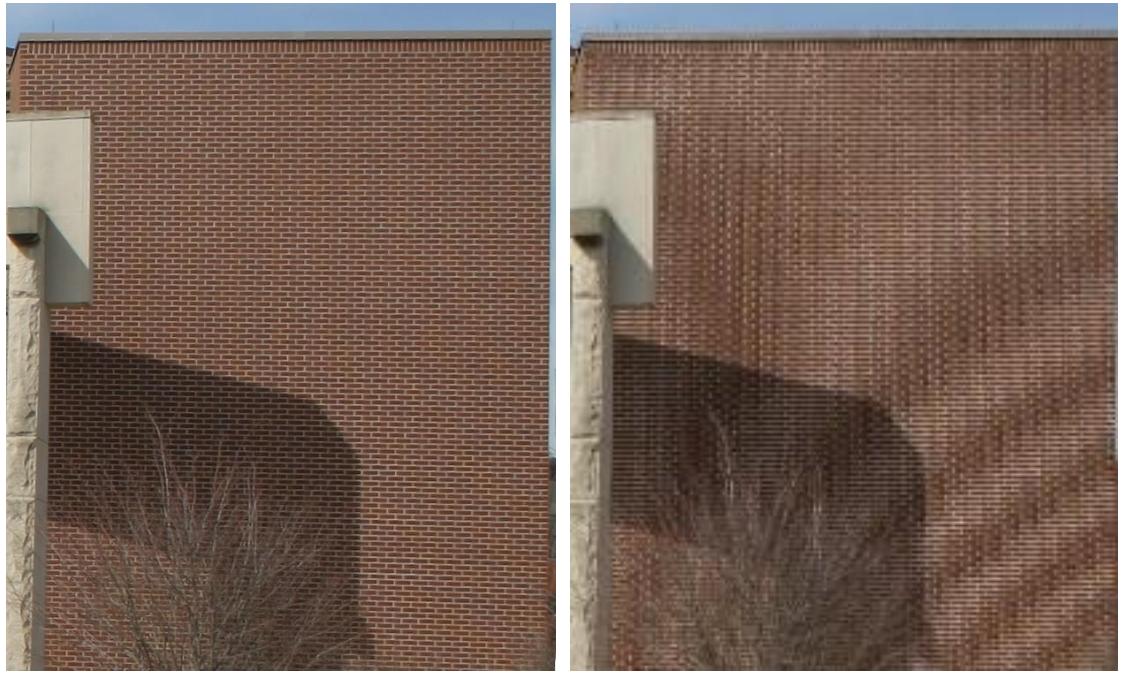
At the frequency level, what are we doing here?

🤔

Source: S. Seitz

Image sub-sampling

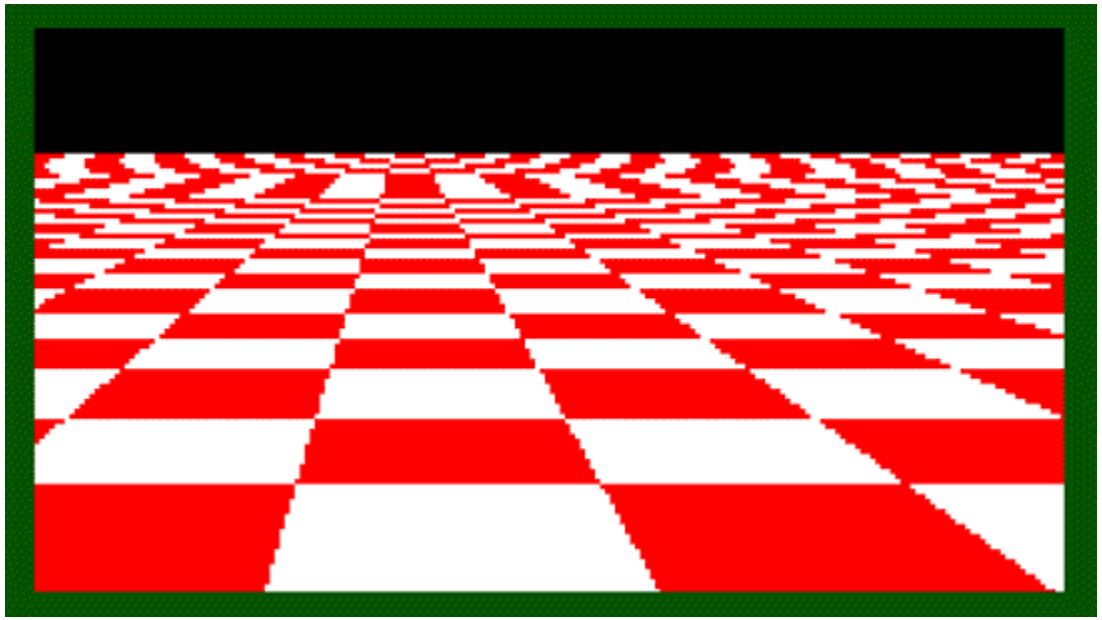
Source: F. Durand



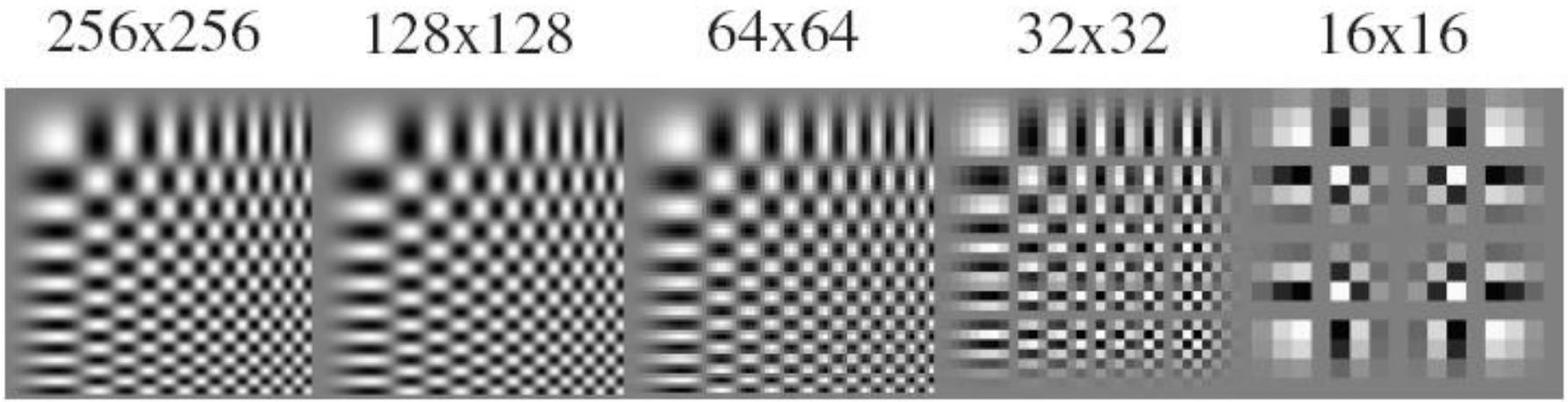
Source: https://en.wikipedia.org/wiki/Aliasing

Aliasing artifacts usually occur in the form of Moiré patterns

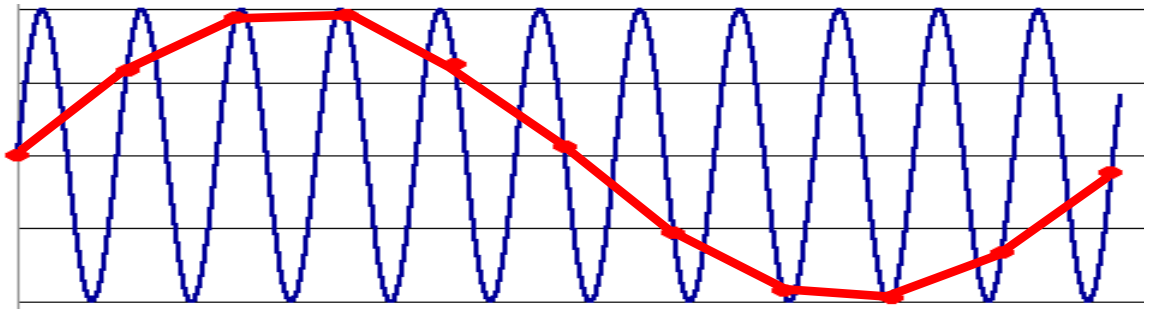
Even worse for synthetic images



Disintegrating textures



Aliasing



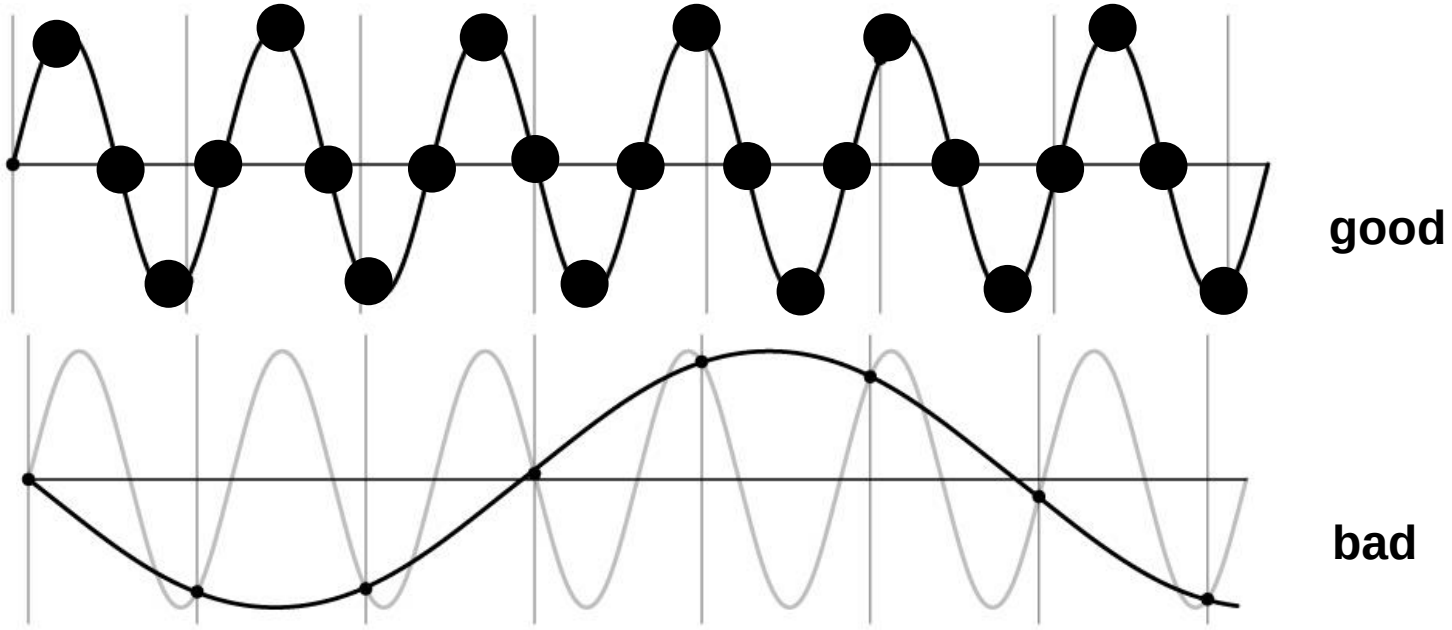
- **Occurs when <u>your sampling rate (either temporal or spatial) is not high enough</u> to capture the amount of detail in your image**
- **Can give you the wrong signal/image**—an *alias*

- To do sampling right, need to understand the structure of your signal/image
  - We need to think in frequency terms!

Aliasing

- **Occurs when <u>your sampling rate (either temporal or spatial) is not high enough</u> to capture the amount of detail in your image**
- **Can give you the wrong signal/image**—an *alias*

- To avoid aliasing:
  - Sample more often!!!
    - sampling rate ≥ 2 * max frequency in the image
      - said another way: ≥ two samples per cycle
    - This minimum sampling rate is called the **Nyquist rate**



Aliasing

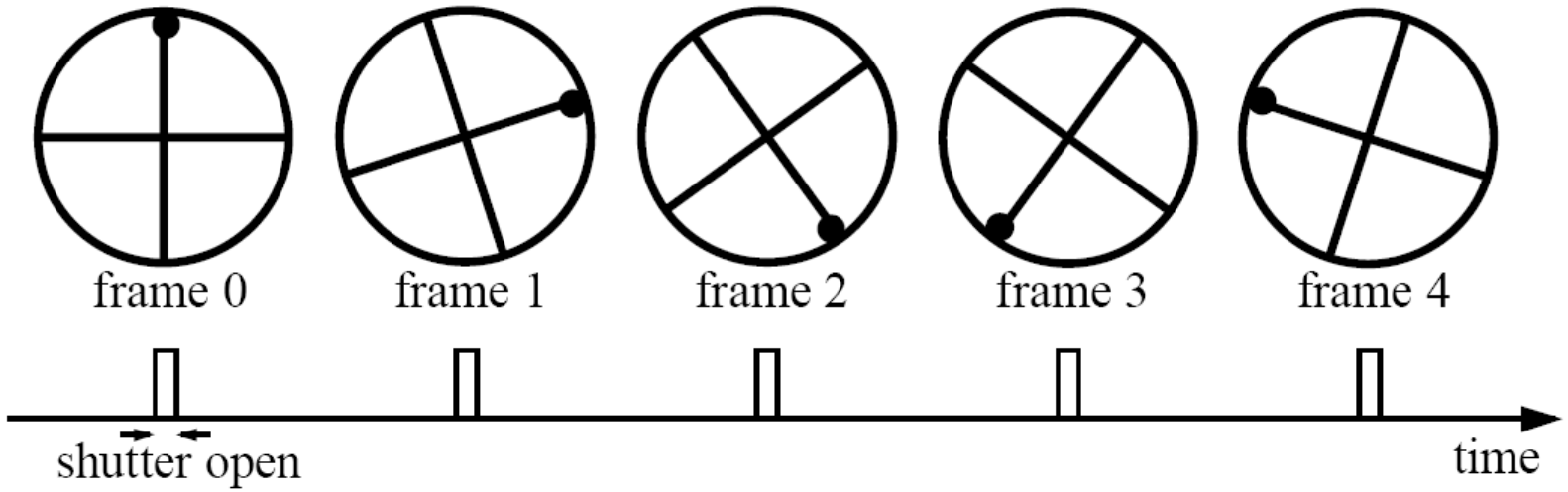
- **Occurs when <u>your sampling rate (either temporal or spatial) is not high enough</u> to capture the amount of detail in your image**
- **Can give you the wrong signal/image**—an *alias*

- To avoid aliasing:
  - Sample more often!!!
    - sampling rate ≥ 2 * max frequency in the image
      - said another way: ≥ two samples per cycle
    - This minimum sampling rate is called the **Nyquist rate**
  - Reduce high-frequencies in the image!!!
    - **Apply a smoothing filter**

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)

This effect is the result of temporal aliasing!

The first examples we saw were the result of spatial aliasing!

(See http://www.michaelbach.de/ot/mot-wagonWheel/index.html)

Source: L. Zhang

Wagon-wheel effect

https://en.wikipedia.org/wiki/Wagon-wheel_effect

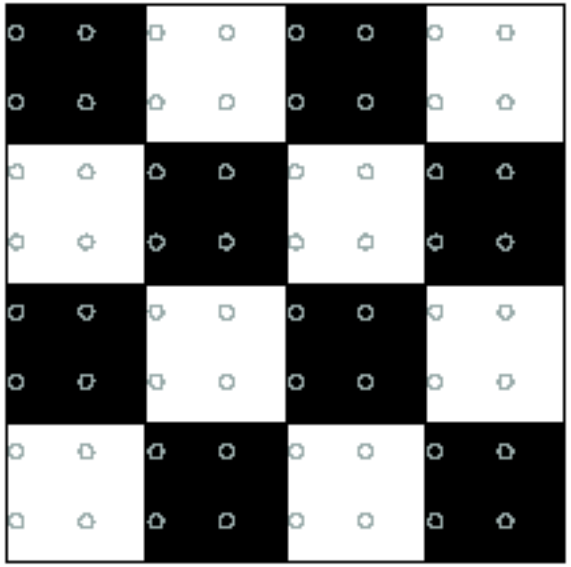
Temporal aliasing – helicopter blades

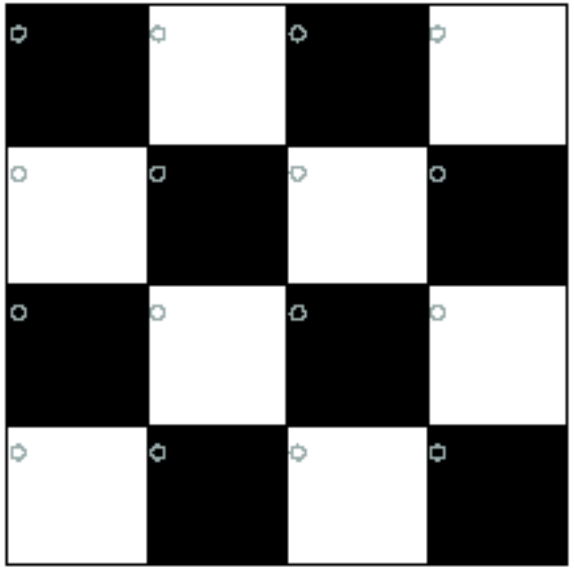
https://www.youtube.com/watch?v=yr3ngmRuGUc

Aliasing in practice

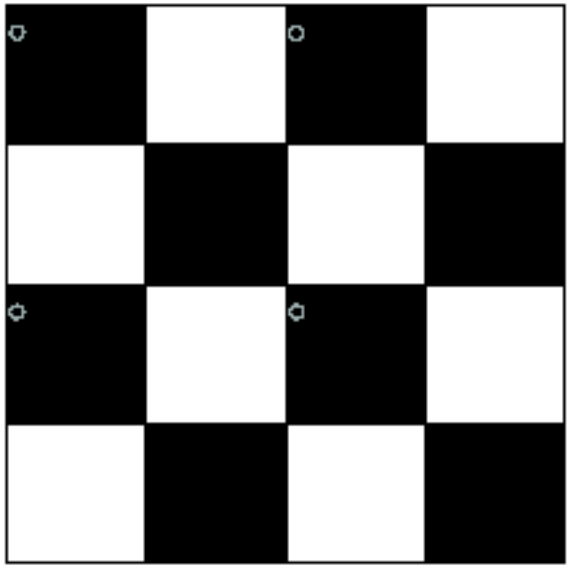
- Sub-sampling may be dangerous….
- Characteristic errors (distortion artifacts) may appear:
  - “Wagon wheels rolling the wrong way in movies”
  - “Checkerboards disintegrate in ray tracing”
  - “Striped shirts look funny on color television”

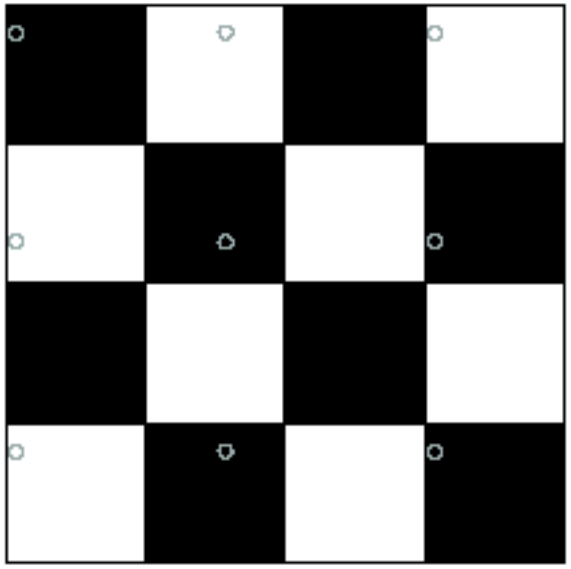
Nyquist limit – 2D example





Good sampling





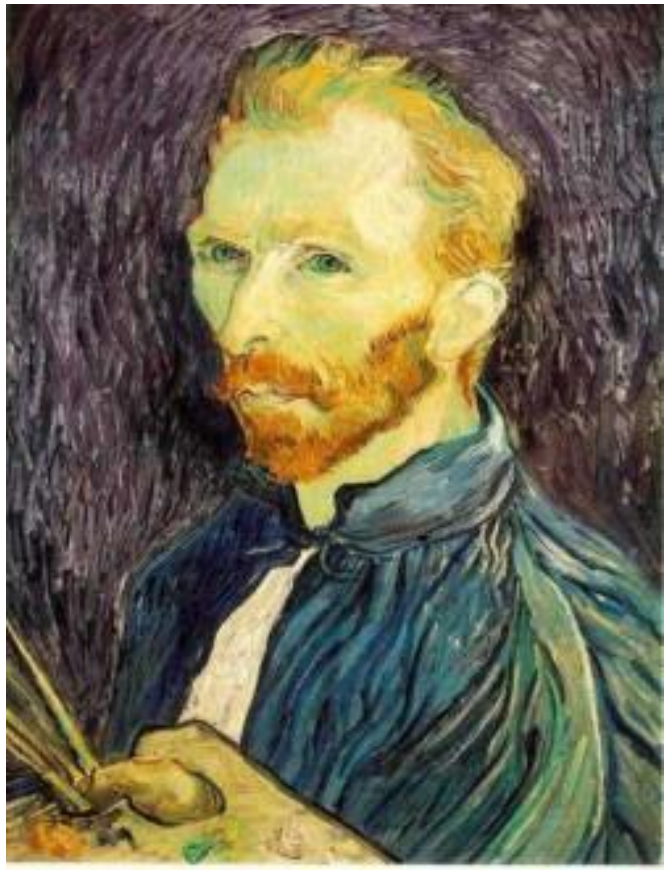
Bad sampling

Aliasing

- When downsampling by a factor of two
  - Original image has frequencies that are too high

- How can we fix this?
  - We need spatial anti-aliasing!

Gaussian pre-filtering



Gaussian 1/2

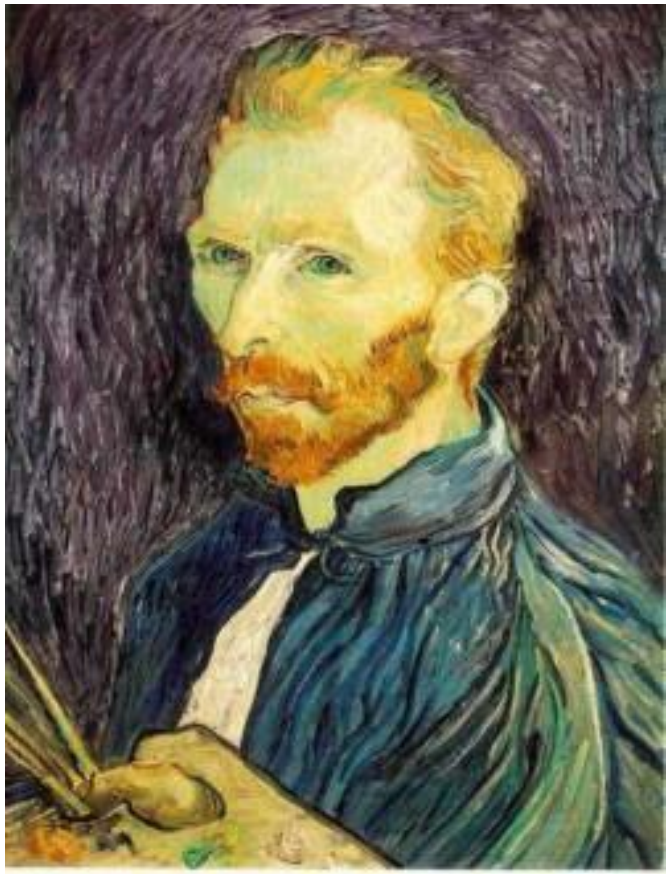
G 1/4

G 1/8

- **Solution:** filter the image, *then* subsample

Source: S. Seitz

Subsampling with Gaussian pre-filtering



Gaussian 1/2

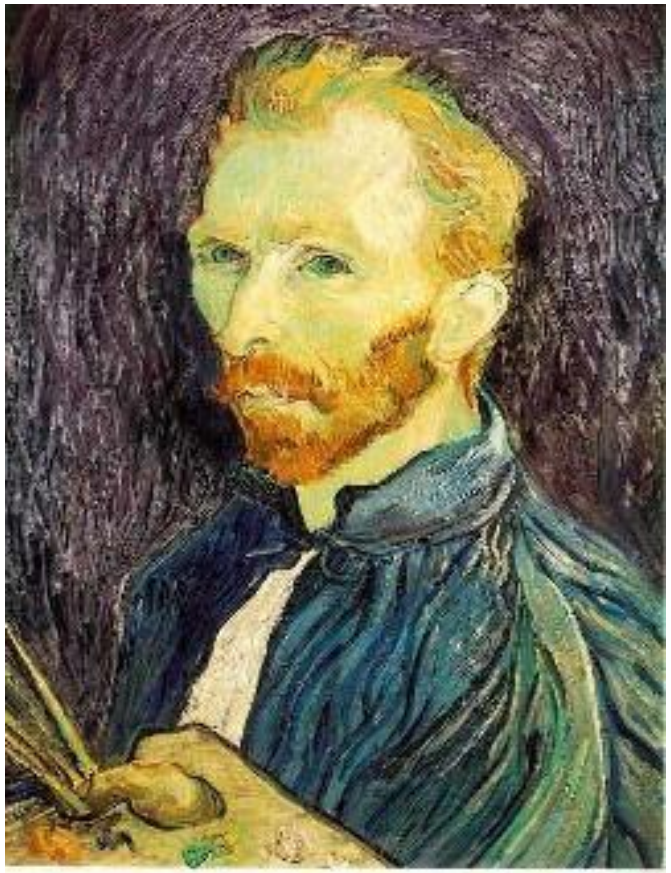
G 1/4

G 1/8

- **Solution:** filter the image, *then* subsample

Source: S. Seitz

Compare with...



1/2

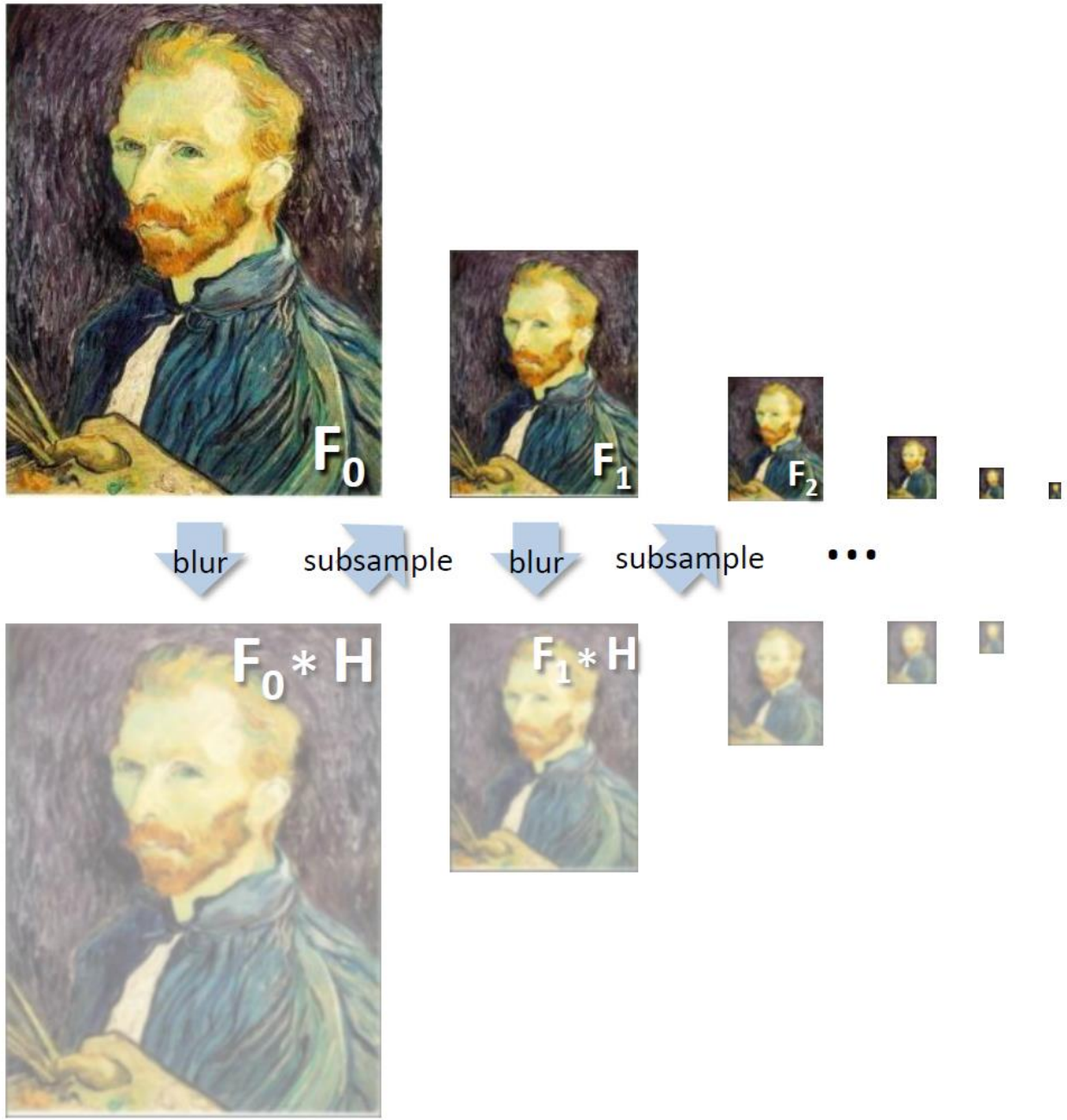
1/4 (2x zoom)

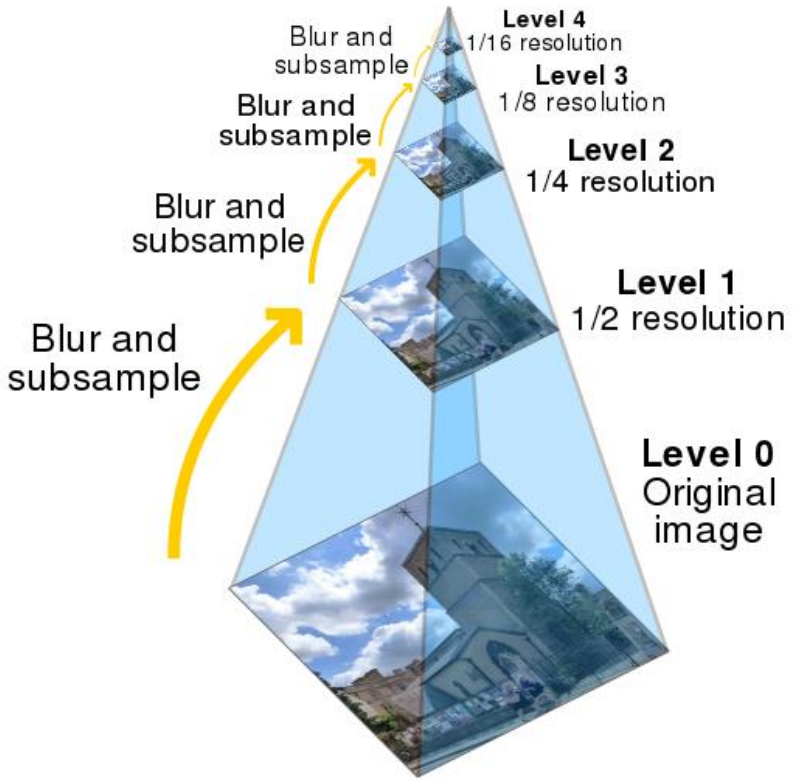
1/8 (4x zoom)

Source: S. Seitz

- **Solution:** filter the image, *then* subsample

Gaussian pyramid

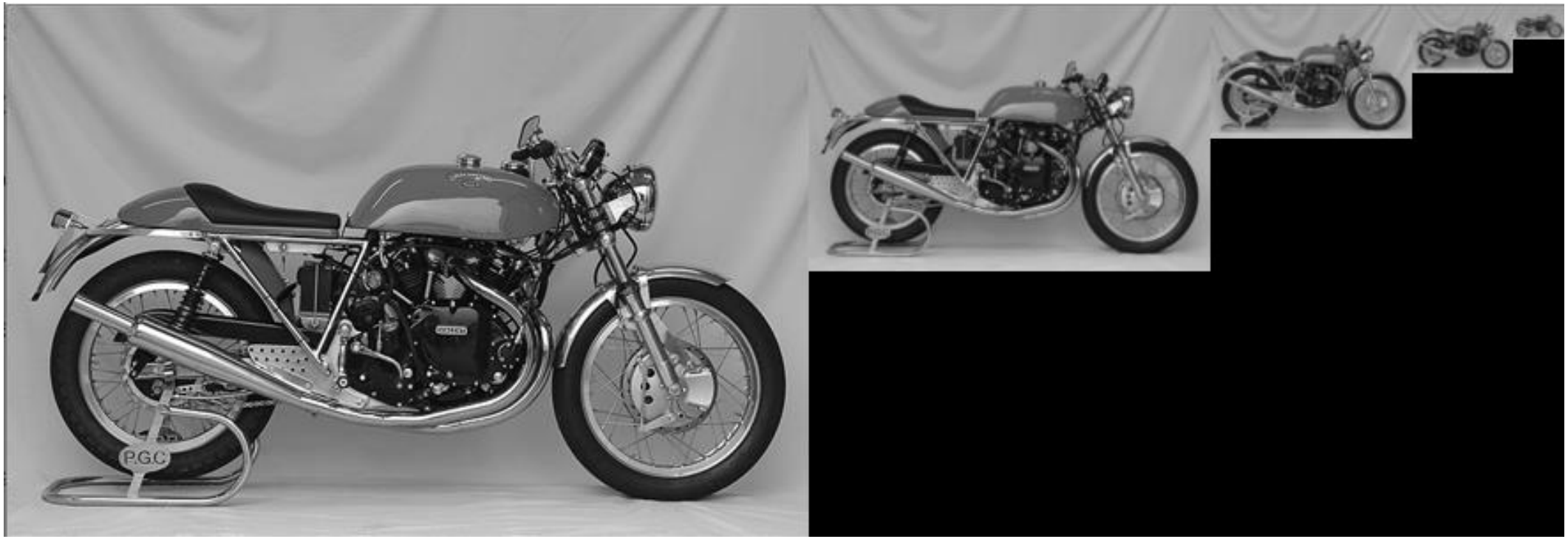




When downsampling an image, **it is common to apply a low-pass filter to the image prior to resampling**.

- We try to avoid that spurious high-frequency information appears in the downsampled image (*aliasing*).

Gaussian pyramids [Burt and Adelson, 1983]



- How much space does a Gaussian pyramid take compared to the original image? 🤔

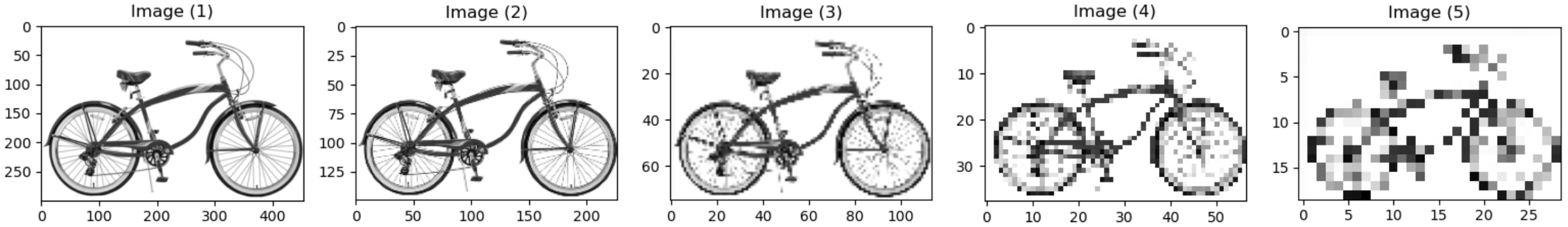
$$1 + \frac{1}{4} + \frac{1}{16} + \cdots + \left(\frac{1}{4}\right)^n$$

As n approaches infinity, the total space approaches 4/3 times the original image size.

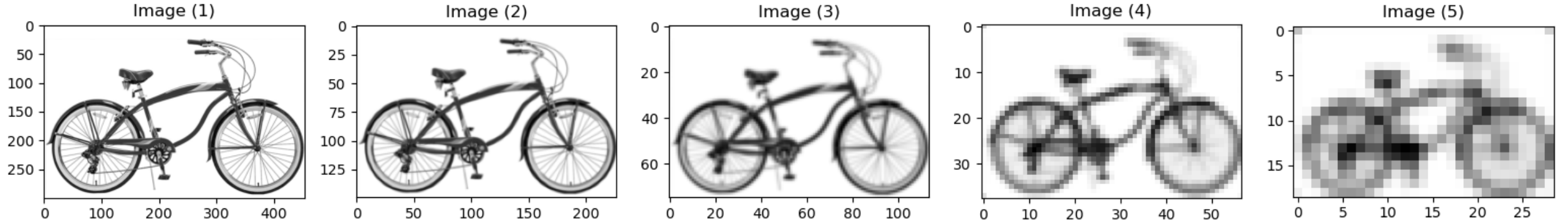
Gaussian Pyramids

- Effect of smoothing when creating the pyramid

No Gaussian smoothing. Just subsampling, keeping the even rows/columns (aliasing):

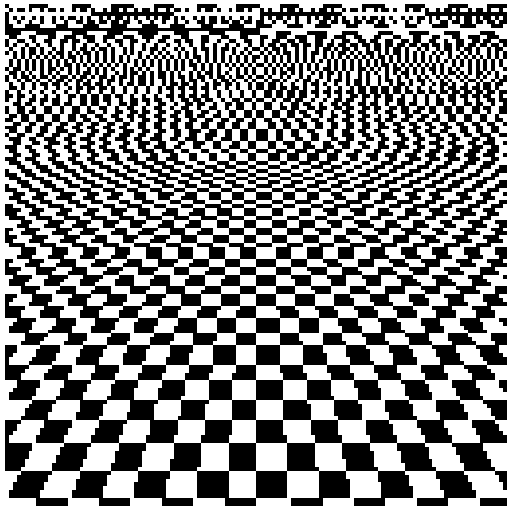


Including Gaussian smoothing:

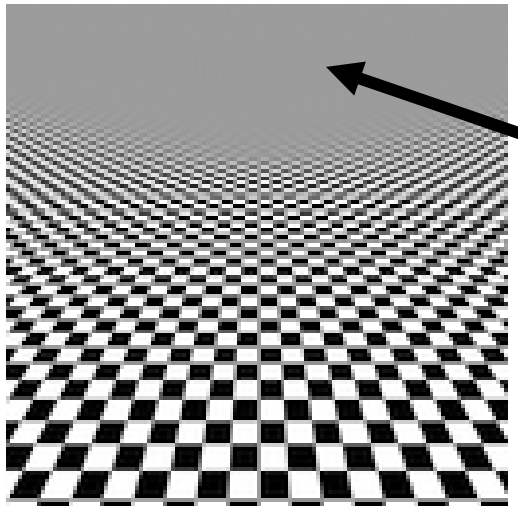


Back to the checkerboard

- What should happen when you make the checkerboard smaller and smaller?



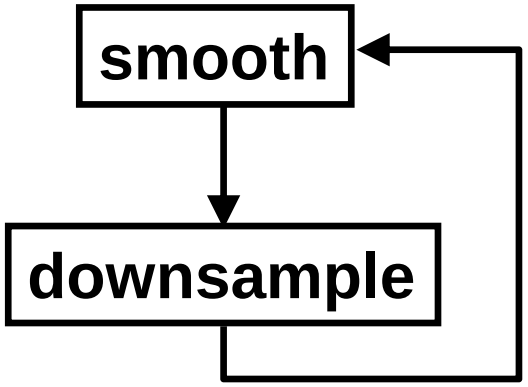
Naïve subsampling

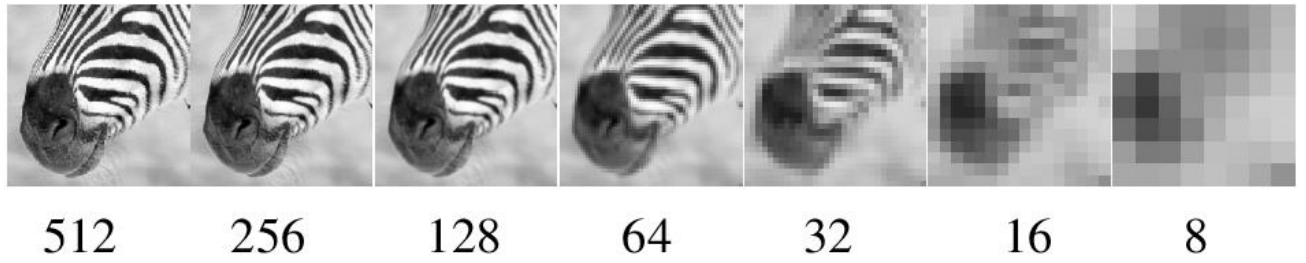


Proper prefiltering (“antialiasing”)

Image turns grey! (Average of black and white squares, because each pixel contains both)

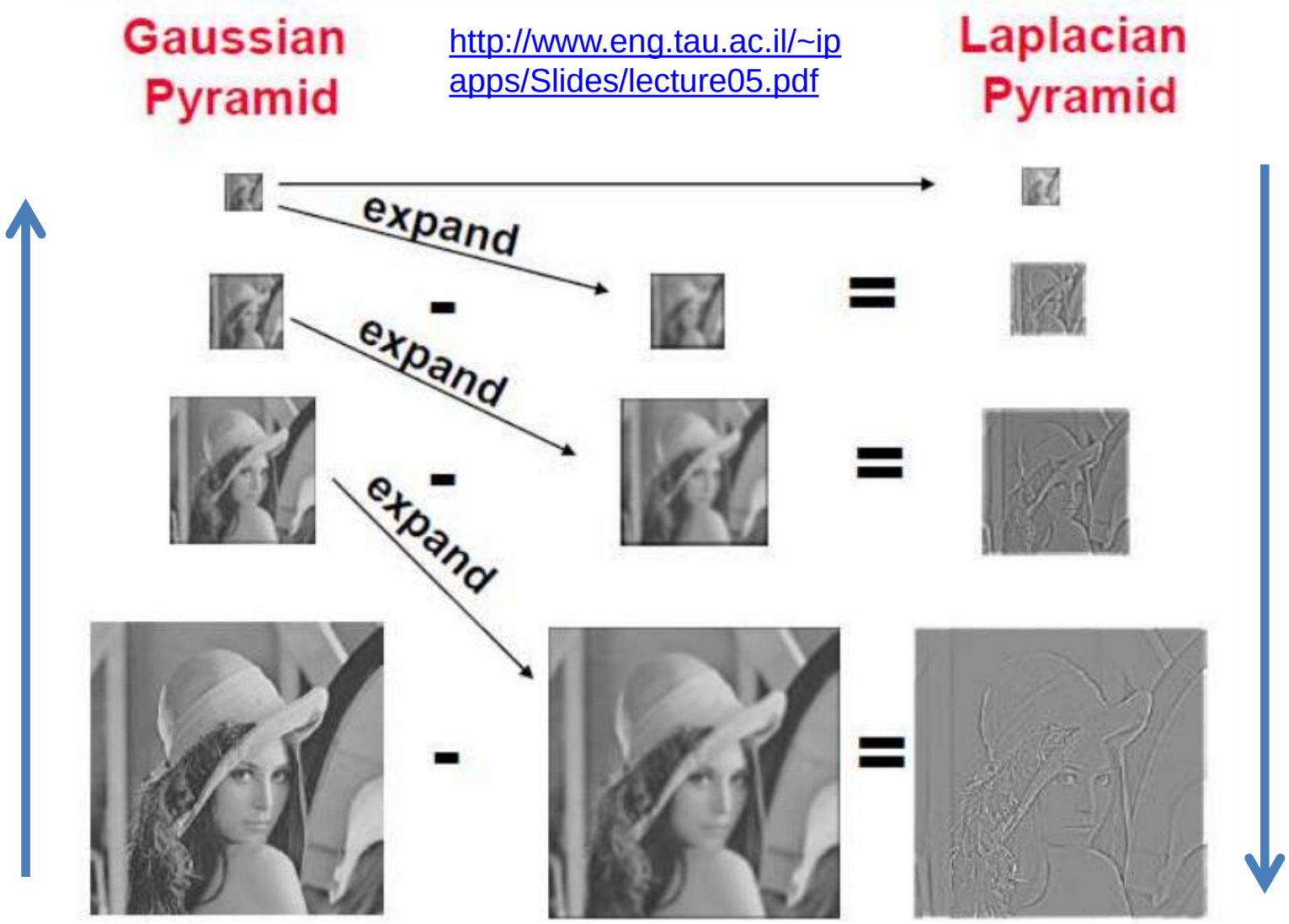
Gaussian Pyramids





Shannon's sampling theorem: After smoothing, many pixels are redundant. Therefore, we can discard them (by downsampling) without losing information

Laplacian Pyramids



1) We start from the smallest image generated by the Gaussian Pyramid.

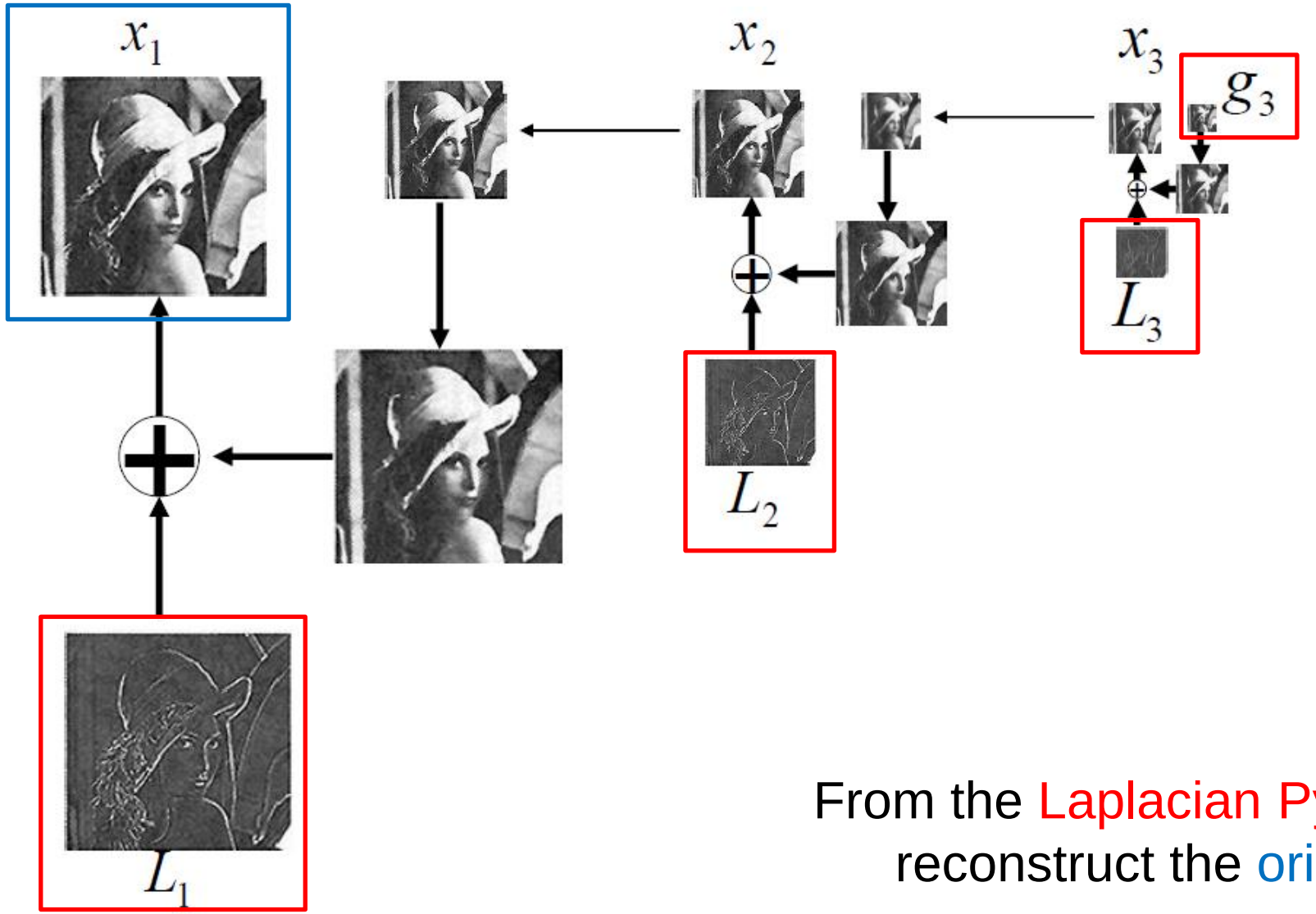
2) We expand it, we compute the difference with the next level in the Gaussian Pyramid, and we get the L_i Level of the Laplacian Pyramid.

3) We move to the next Gaussian level, we expand it, compute the difference, and we already have level L_{i-1} in the Laplacian Pyramid.

4) And so on...

Laplacian Pyramids

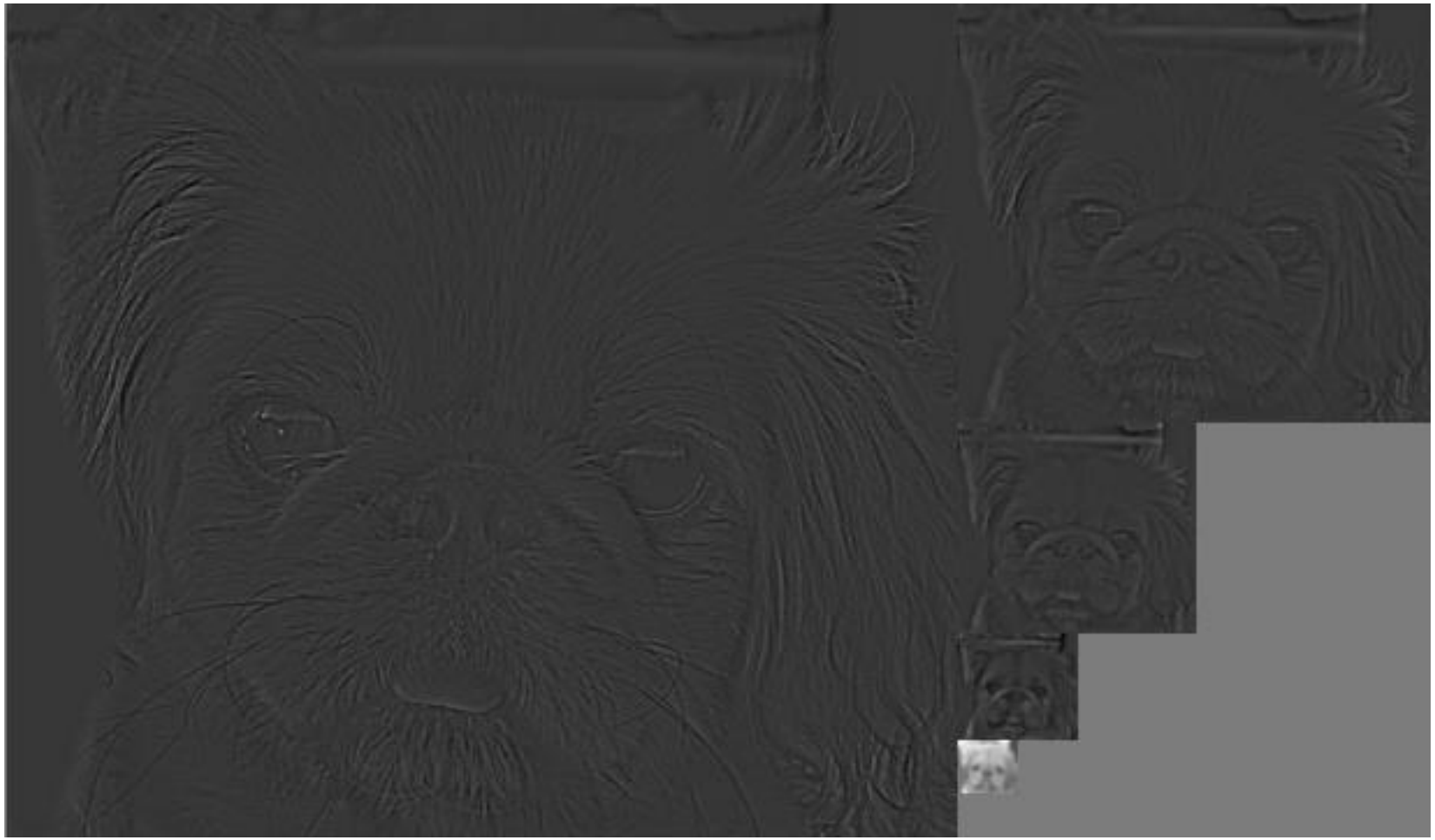
- We are able to reconstruct x_1 from L_1, L_2, L_3 and g_3



From the Laplacian Pyramid, we can reconstruct the original image

Gaussian Pyramid

Laplacian Pyramid



Laplacian Pyramid

Original Image **Reconstructed Image**

All pixels are the same!! We have a reduced version of the original image, and all the details (high frequencies) at different scales. So, we have all the necessary information for a perfect reconstruction!

What are image pyramids used for?

Image compression

Multi-scale texture mapping

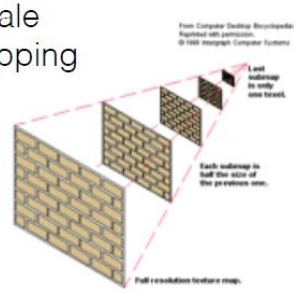
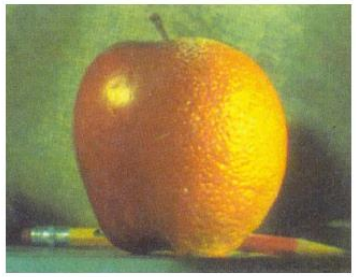
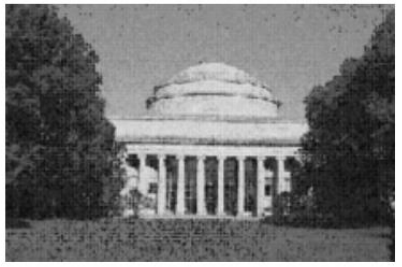


Image blending



Multi-focus composites

Noise removal



Hybrid images

Multi-scale detection

Multi-scale registration

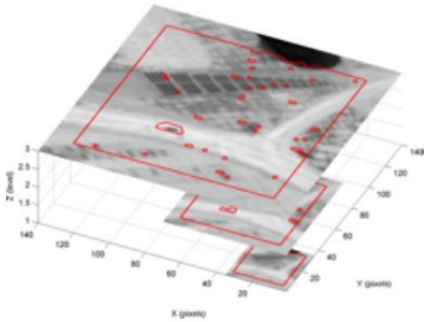


Image Blending

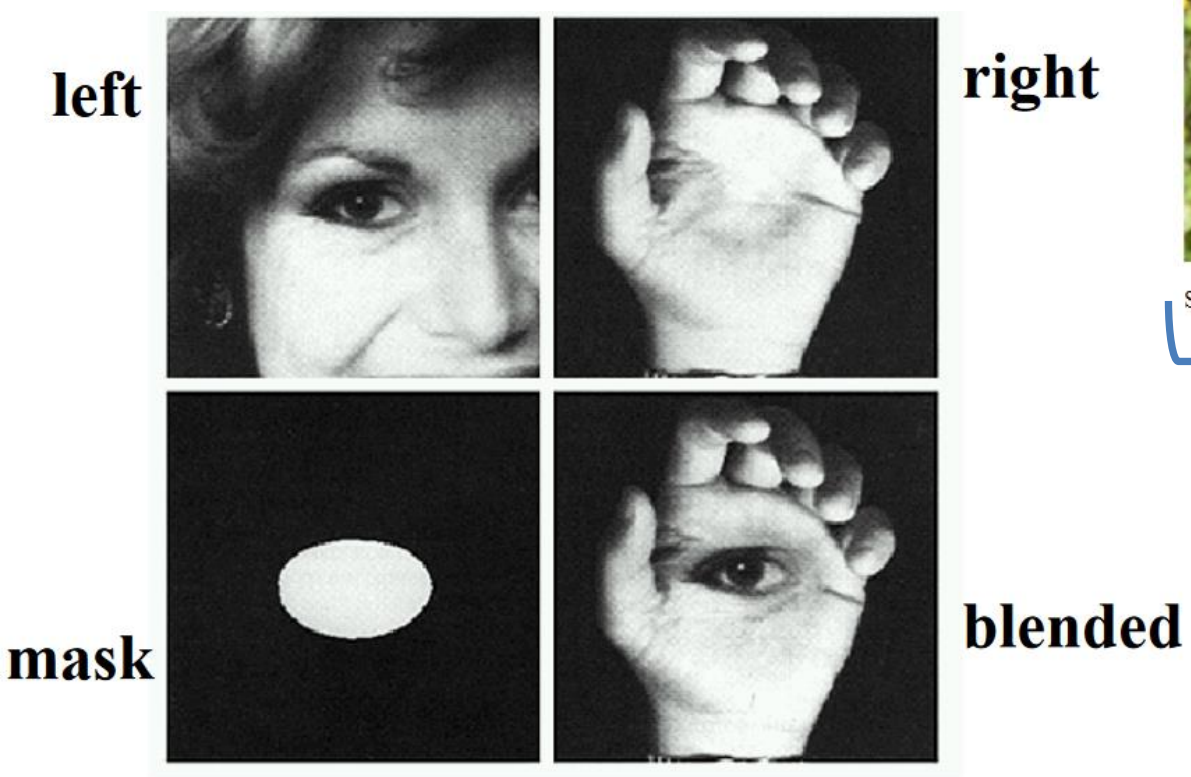
vs

1) Generate the Laplacian pyramid (L_A and L_B) for both input images (A and B)
   - To do so, you first need to computer Gaussian pyramids
2) Build a Gaussian pyramid G_R from the selected region R
   - R: mask that says which pixels come from image A and which from image B
3) Form a combined Laplacian pyramid L from L_A and L_B using nodes/pixels of G_R as weights:

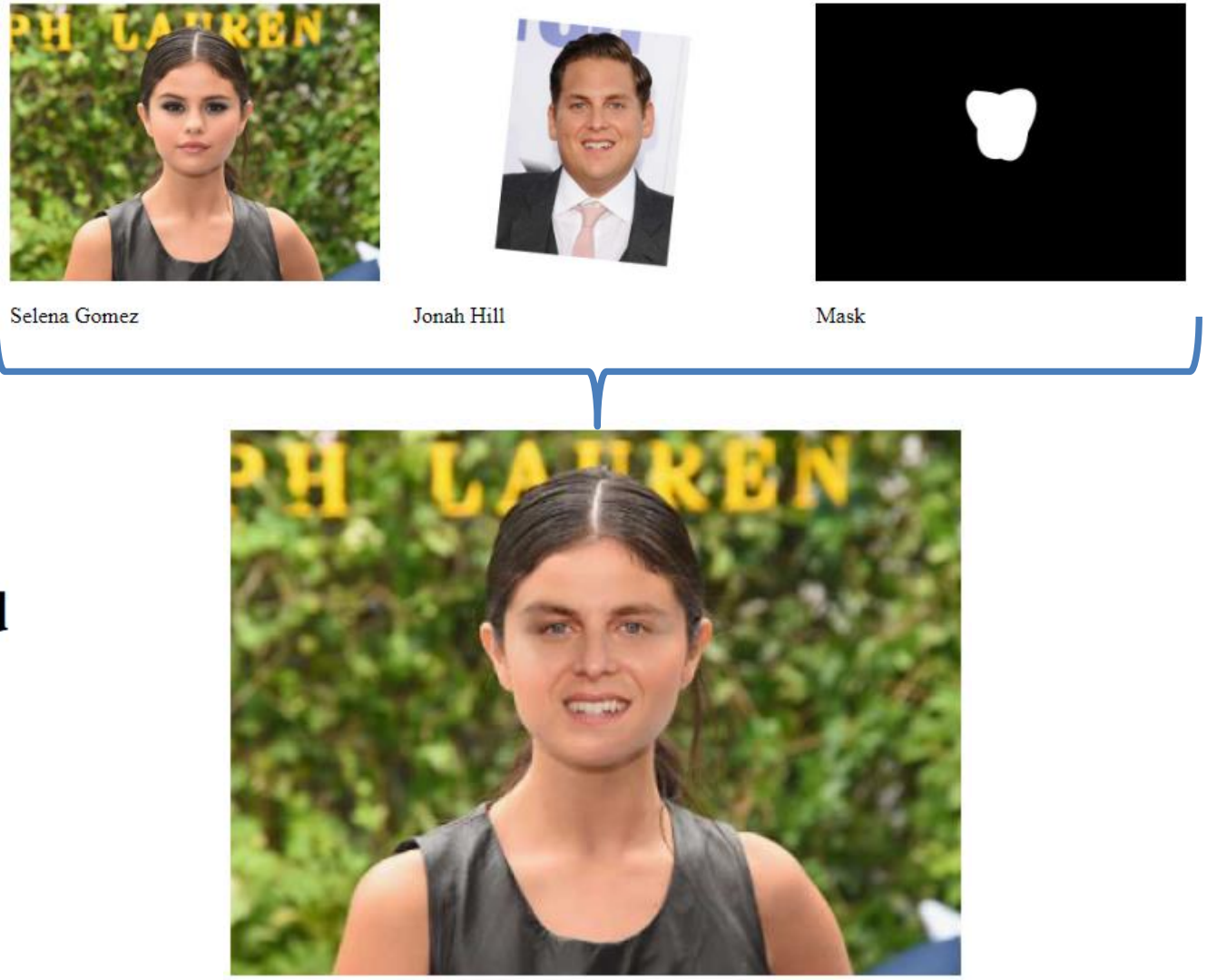
$$L(i,j) = GR(i,j)L_A(i,j) + \big(1 - GR(i,j)\big)L_B(i,j)$$

4) Collapse this new mixed/hybrid Laplacian pyramid, and reconstruct the “original” image

Image Blending



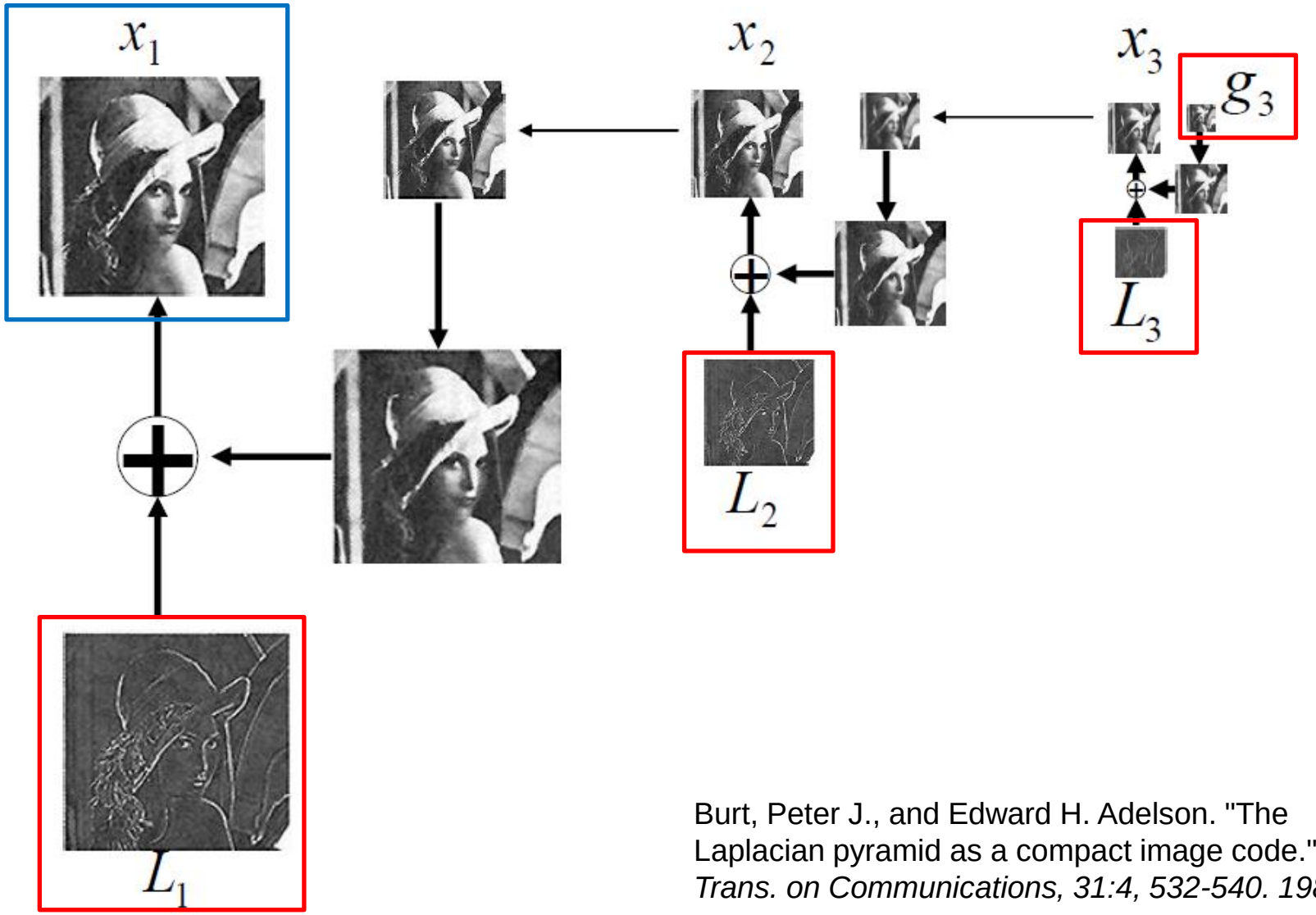
Burt and Adelson, “A multiresolution spline with application to image mosaics,” ACM Transactions on Graphics, 1983, Vol.2, pp.217-236.



https://inst.eecs.berkeley.edu/~cs194-26/fa17/upload/files/proj3/cs194-26-abq/

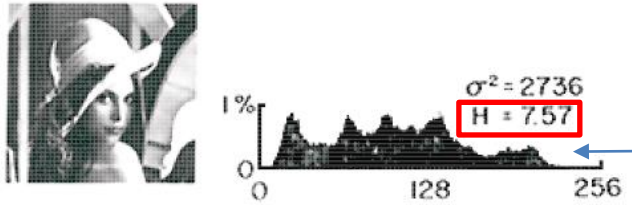
Image Compression

- To reconstruct the original image (x_1), we just need to store L_1, L_2, L_3 and g_3

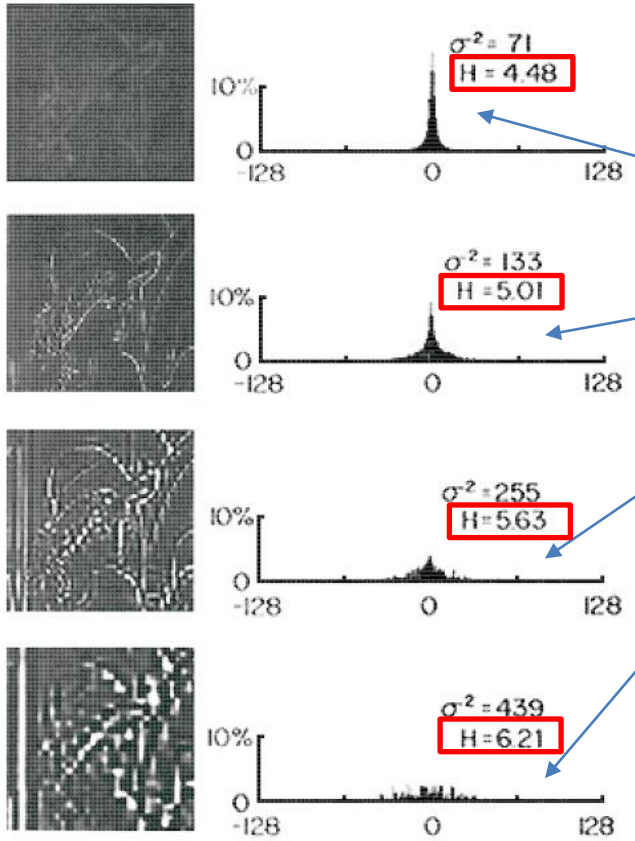


Burt, Peter J., and Edward H. Adelson. "The Laplacian pyramid as a compact image code." *IEEE Trans. on Communications, 31:4, 532-540. 1983.*

Image Compression



Histogram of the original image



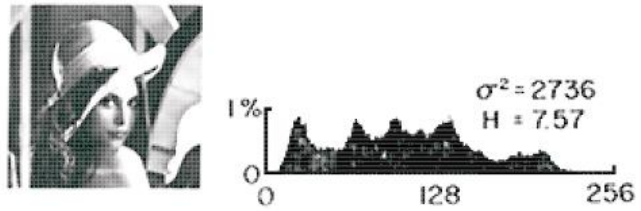
Distribution of pixel gray level values at various stages of the encoding process.

Pixel values in the Laplacian pyramid are concentrated near zero

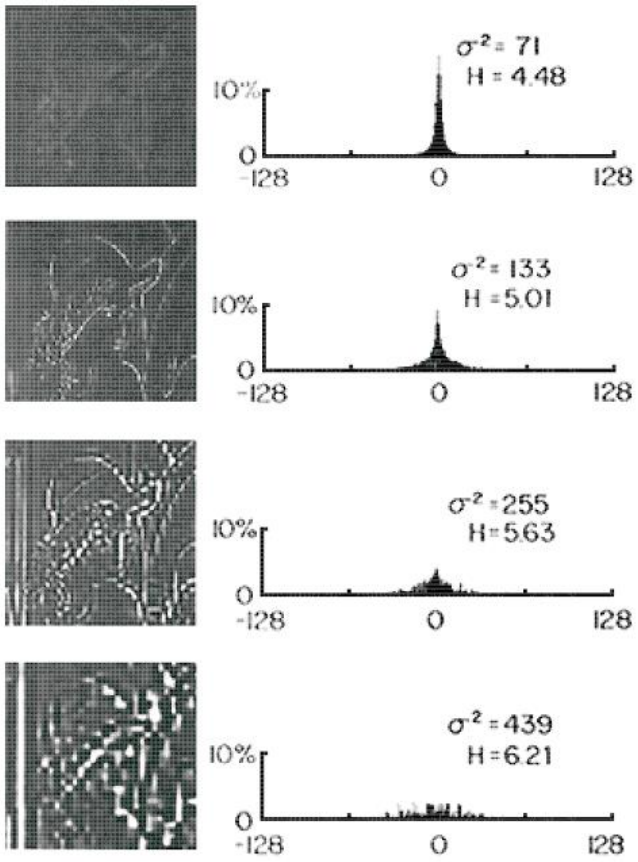
Histogram entropy represents the minimum number of bits per pixel required to exactly encode the image. The maximum entropy would be 8, since the image is initially represented at 256 gray levels, and would be obtained when all gray levels were equally likely.

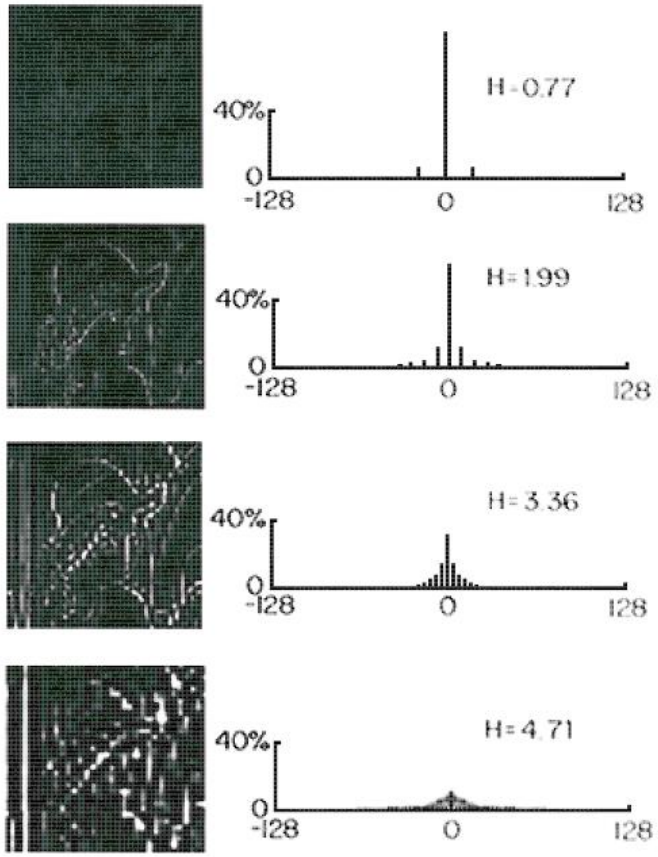
Burt, Peter J., and Edward H. Adelson. "The Laplacian pyramid as a compact image code." *IEEE Trans. on Communications, 31:4, 532-540. 1983.*

Image Compression



Substantial further reduction is obtained through quantization





4.4 vs 0.77

5.0 vs 1.9

5.6 vs 3.3

6.2 vs 4.7

Burt, Peter J., and Edward H. Adelson. "The Laplacian pyramid as a compact image code." *IEEE Trans. on Communications, 31:4, 532-540. 1983.*

Image Compression

ORIGINAL **COMPRESSED**

(a)

(b)

1.58 bits/pixel

The encoded images are almost indistinguishable from the originals

(c)

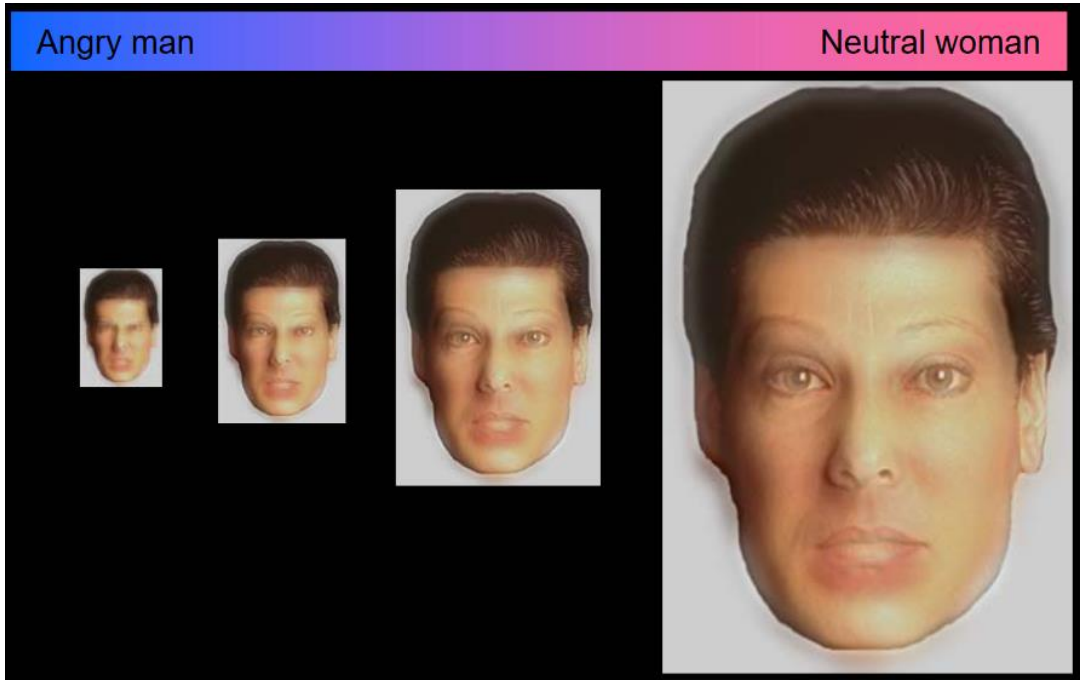
(d)

0.73 bits/pixel

Burt, Peter J., and Edward H. Adelson. "The Laplacian pyramid as a compact image code." *IEEE Trans. on Communications, 31:4, 532-540. 1983.*

Hybrid Images

A. Oliva, A. Torralba, P.G. Schyns (2006). Hybrid Images. ACM Transactions on Graphics. http://olivalab.mit.edu/hybridimage.htm



- By properly **mixing** a part of the **high frequencies of one image with** a part of the **low frequencies of another image,** we obtain a **hybrid image** that **allows different interpretations at different distances.**

We evaluate the final result using the Gaussian Pyramid!!

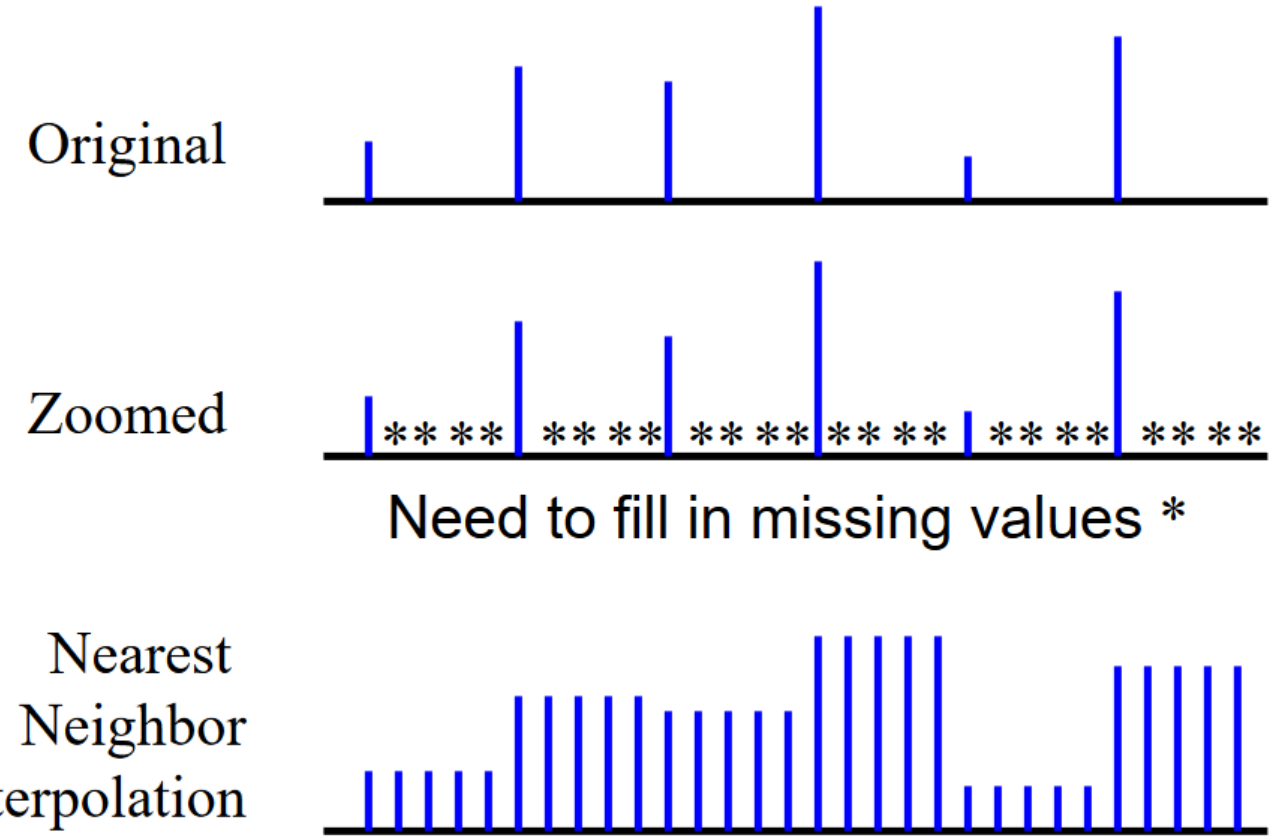
Low frequencies prevail at **long distances**, while
high frequencies prevail at **short distances**.

Upsampling

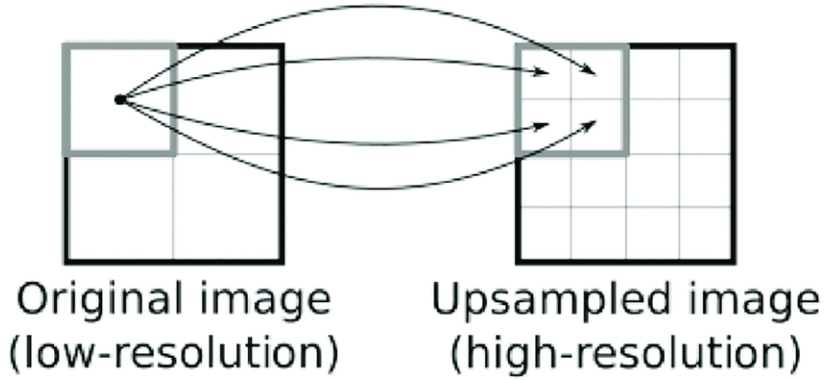
- This image is too small for this screen: 
- How can we make it 10 times as big?
- Simplest approach:
  repeat each row
  and column 10 times
  + “Nearest neighbor
  interpolation”

Nearest Neighbor Interpolation

1D

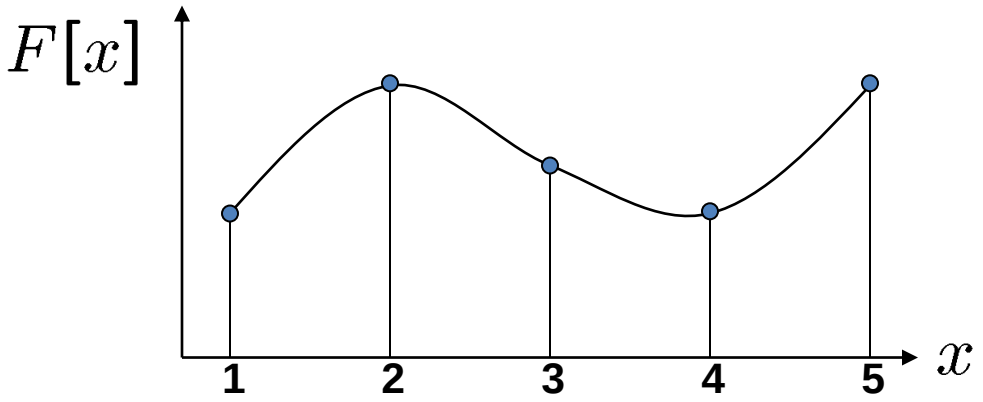


2D



Source: R. Rao

Image interpolation



Recall that a digital image is formed as follows:

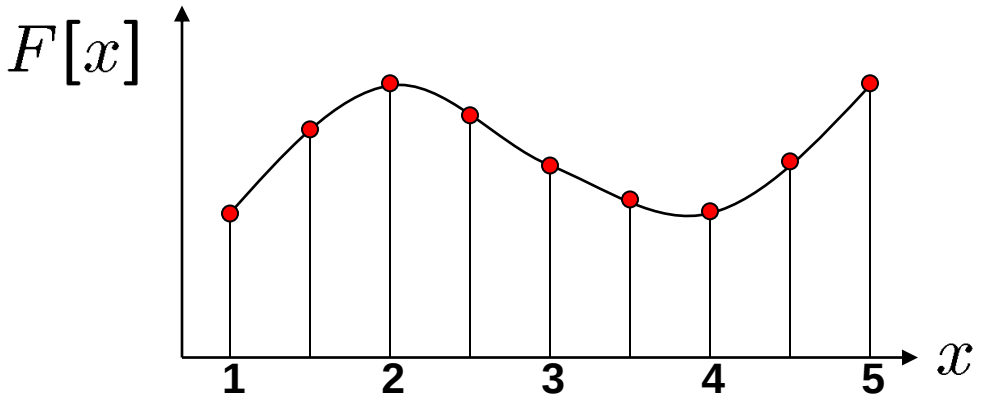
$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$

d = 1 in this example

- 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



Recall that a digital image is formed as follows:

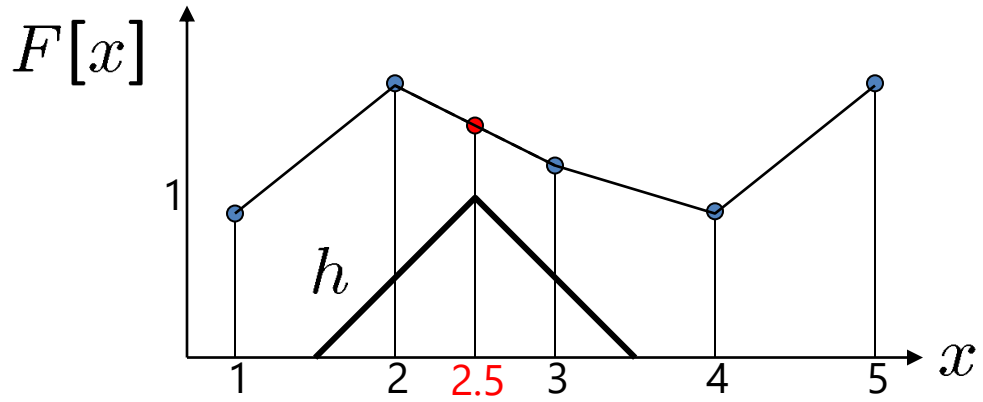
$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$

d = 1 in this example

- 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

- But… we don't know f 😔
  - Guess an approximation: $\tilde{f}$
  - Can be done in a **principled way: filtering**
  - Convert F to a continuous function:

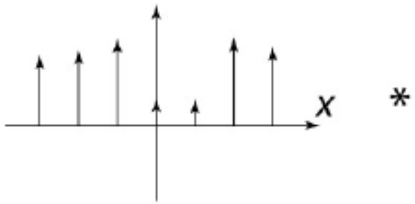
    $f_F(x) = F(\frac{x}{d})$ when $\frac{x}{d}$ is an integer, 0 otherwise

  - **Reconstruct by convolution with a *reconstruction filter*, h**

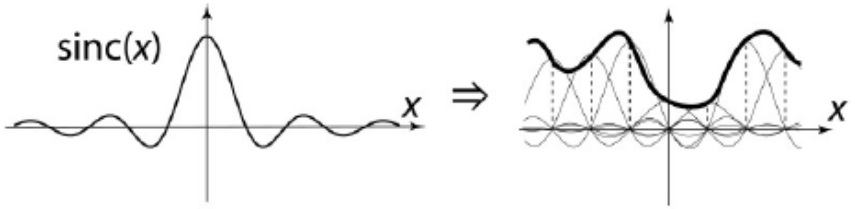
$$\tilde{f} = h * f_F$$

Adapted from: Snavely

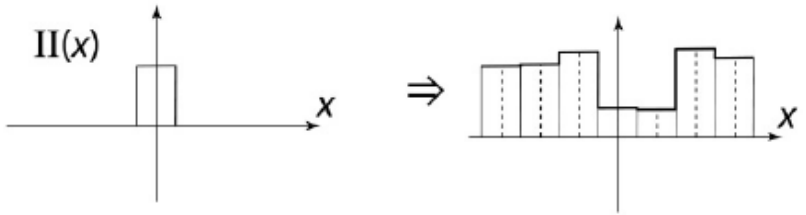
Image interpolation



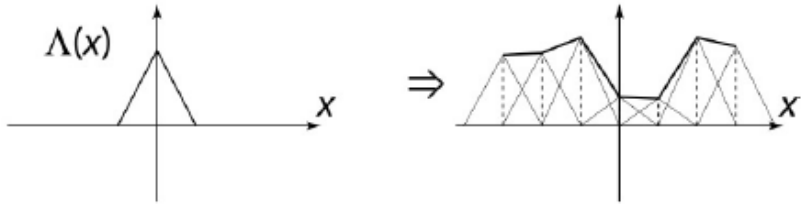
*



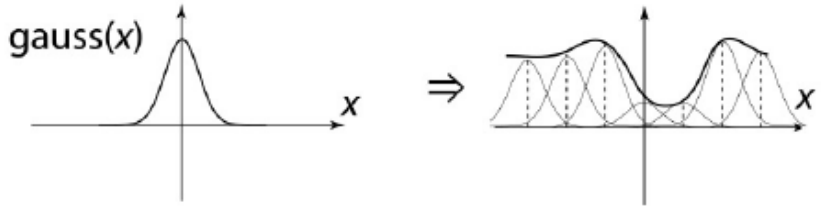
Reconstruction using the sinc function



Nearest-neighbor interpolation



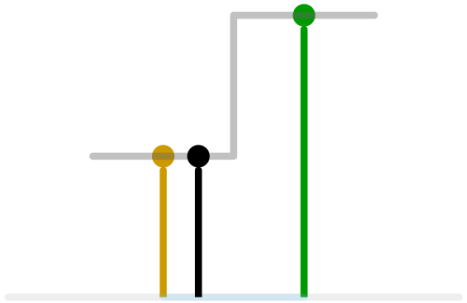
Linear interpolation



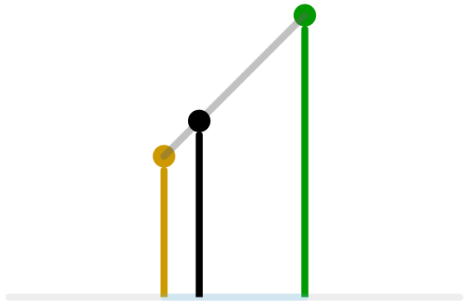
Gaussian reconstruction

Source: B. Curless

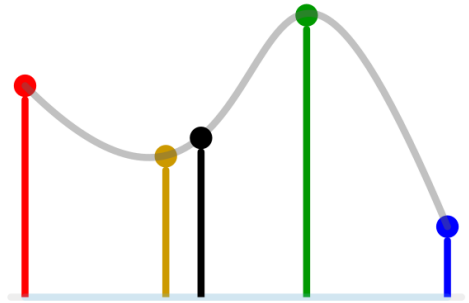
Image interpolation



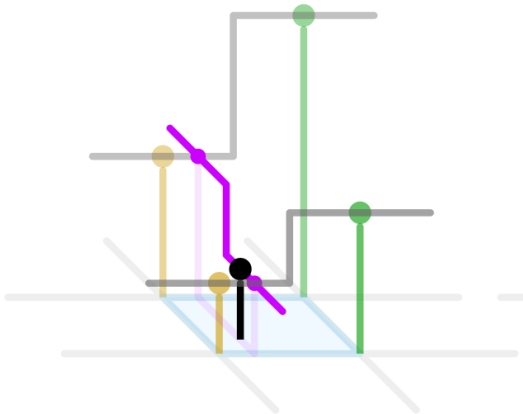
1D nearest-neighbour



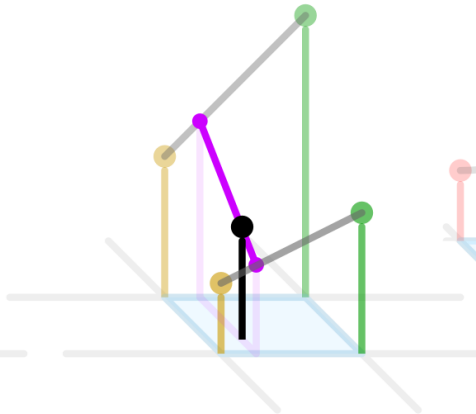
Linear



Cubic

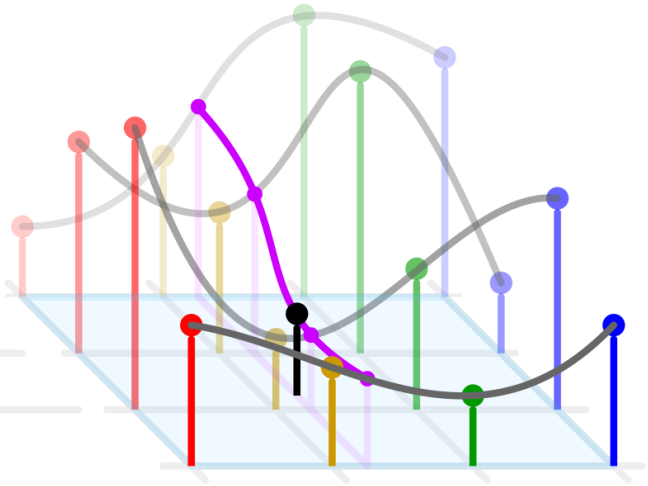


2D nearest-neighbour



Bilinear

It considers 4 nearest pixels (2×2)



Bicubic

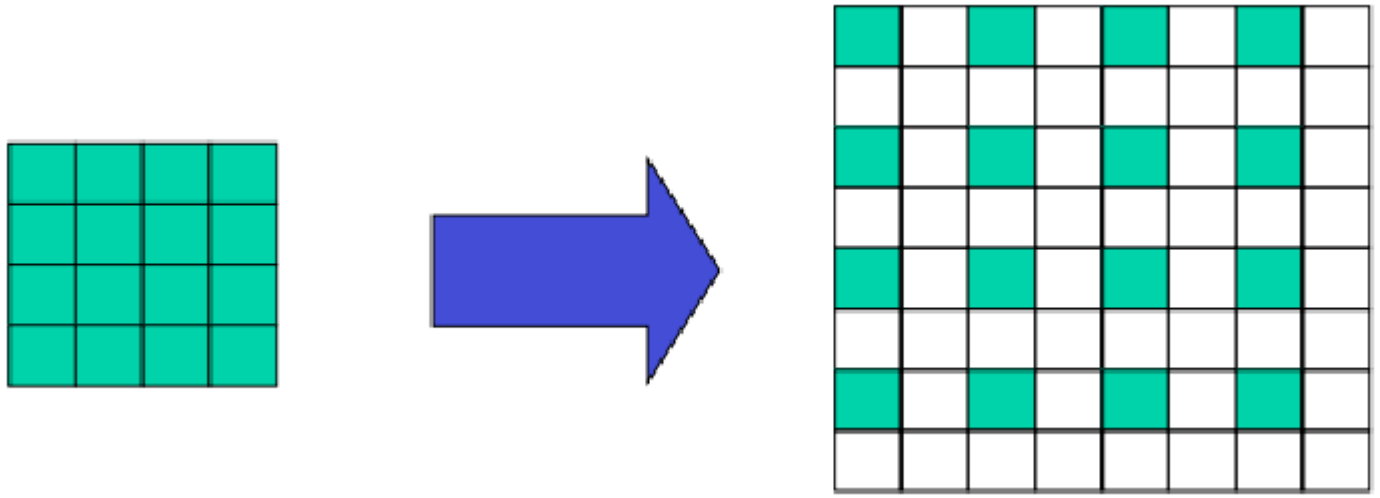
It considers 16 nearest pixels (4×4)

Deep learning approaches are also successfully used in image scaling. See Wang, Xintao, et al. “ESRGAN: Enhanced super-resolution generative adversarial networks." *Proceedings of the European conference on computer vision (ECCV) workshops*. 2018.

Black and red/yellow/green/blue dots correspond to the interpolated point and neighboring samples, respectively. Their heights above the ground correspond to their values.

https://en.wikipedia.org/wiki/Nearest-neighbor_interpolation#/media/File:Comparison_of_1D_and_2D_interpolation.svg

Upsampling



How to fill in the emply values?

- The empty pixels are initially set to 0
- Convolve upsampled image with Gaussian filter
- In this example, we must also multiply by 4. Why?? 🤔

Gaussian filter is a smoothing filter. It sums to 1.
¾ of the new image was initially 0. We need to “recover” intensity!

Image interpolation

Original image:  **x 10**

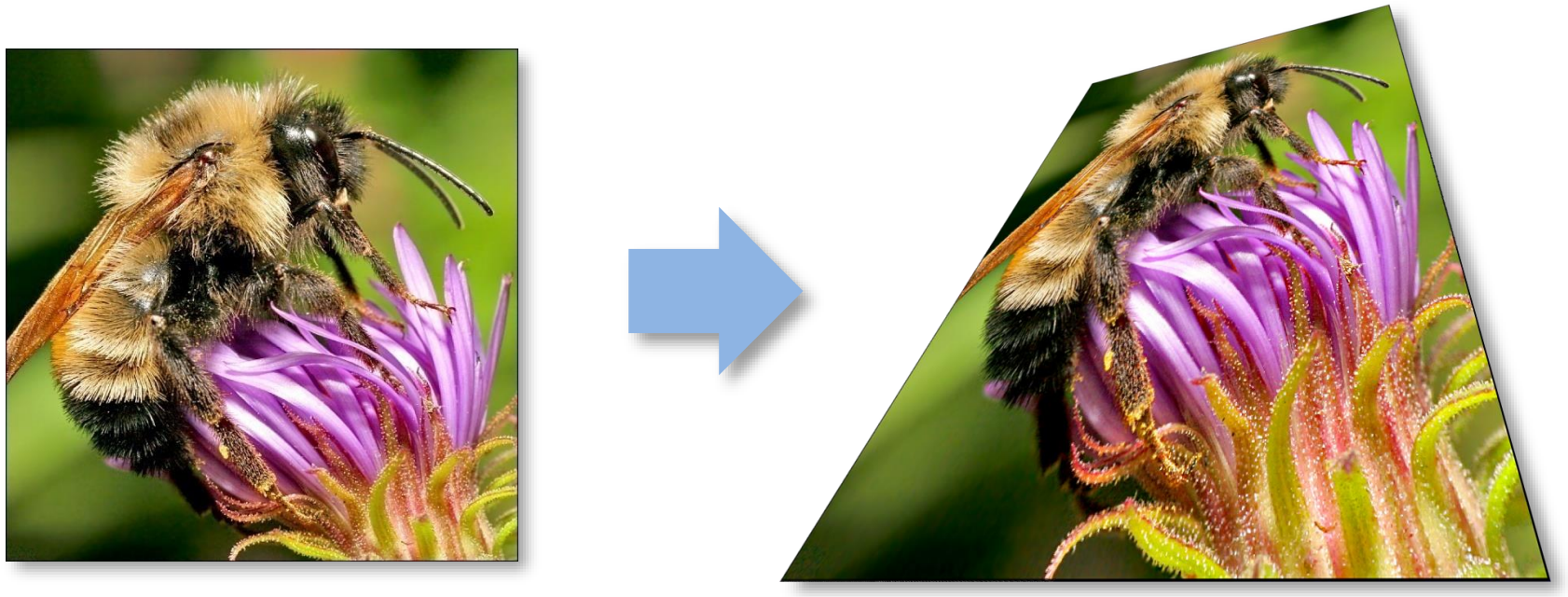
Nearest-neighbor interpolation

Bilinear interpolation

Bicubic interpolation (common choice)

Image interpolation

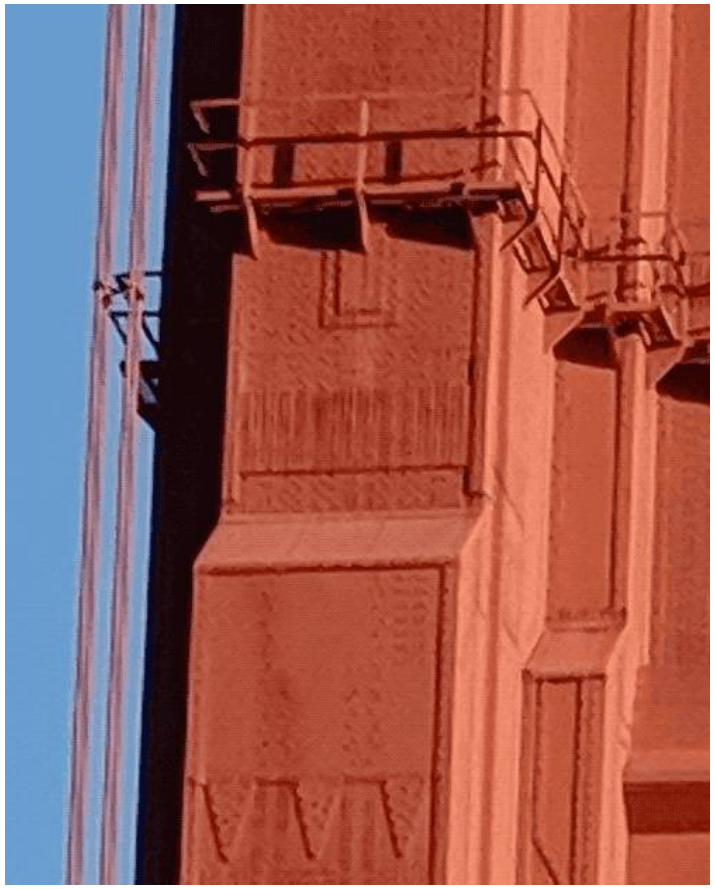
Also used for *resampling*



Super-resolution with multiple images

- Can do better upsampling if you have multiple images of the scene taken with small shifts
- Some cellphone cameras (like the Google Pixel line) capture a burst of photos
- Can we use that burst for upsampling?

Google Pixel 3 Super Res Zoom



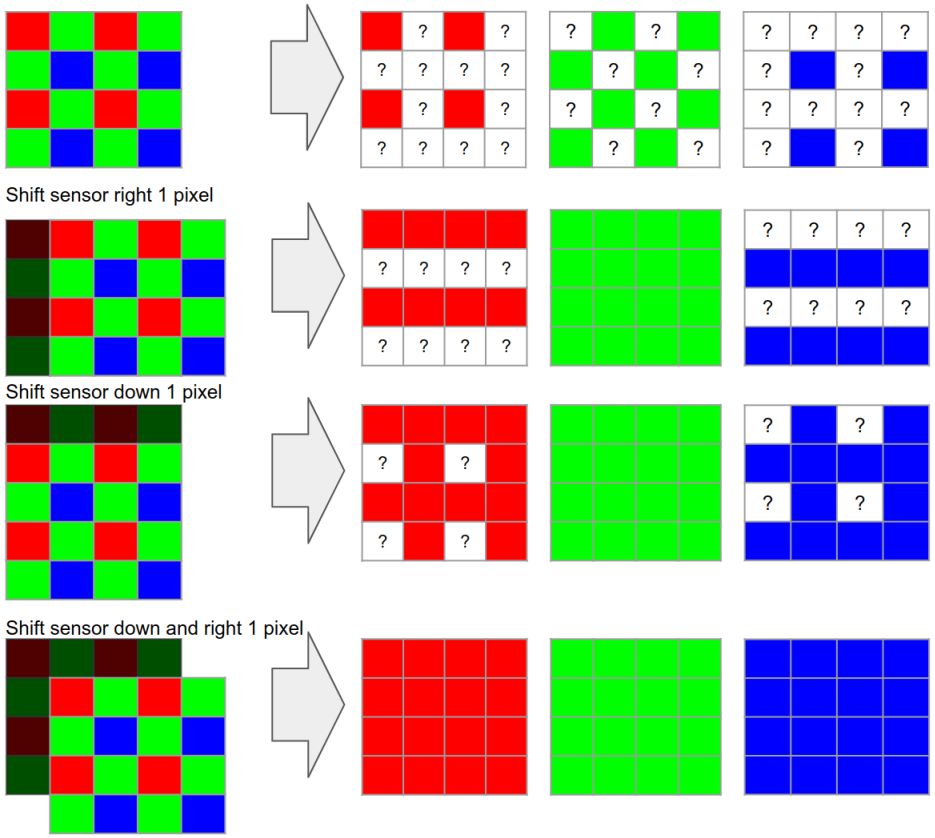
Example photo with and without super res zoom (smart burst align and merge)

This technique merges many frames directly onto a higher resolution picture *(multi-frame super-resolution)*

https://ai.googleblog.com/2018/10/see-better-and-further-with-super-res.html

Google Pixel 3 Super Res Zoom

Example of how an idealized multi-frame super-resolution algorithm might work:



As compared to the standard demosaicing pipeline that needs to interpolate the missing colors (top), ideally, one could fill some holes from multiple images, each shifted by one pixel horizontally or vertically.

We can take advantage of the effect of hand tremor (in this example, as seen in a cropped burst of photos, after global alignment)

https://ai.googleblog.com/2018/10/see-better-and-further-with-super-res.html

Summary

- Key points:
  - **Subsampling an image** can cause aliasing. Better is to blur ("pre-filter") to remove high frequencies then downsample.
  - If you repeatedly blur and downsample by 2x, you get a **Gaussian Pyramid**.
  - From this Gaussian Pyramid, we can compute the **Laplacian Pyramid** (that can be useful in image blending or image compression).
  - **Upsampling an image** requires interpolation. This can be posed as convolution with a "reconstruction kernel".

Pyramids

Pablo Mesejo

pmesejo@go.ugr.es

Universidad de Granada

Departamento de Ciencias de la Computación e Inteligencia Artificial

UNIVERSIDAD DE GRANADA

DaSCI

Instituto Andaluz de Investigación en **Data Science** and **Computational Intelligence**