Filtering and resampling

Yoni Chechik

www.AlisMath.com

References

- http://szeliski.org/Book/
- http://www.cs.cornell.edu/courses/cs5670/2019sp/lectures/lectures.html
- http://www.cs.cmu.edu/~16385/

Some motivation

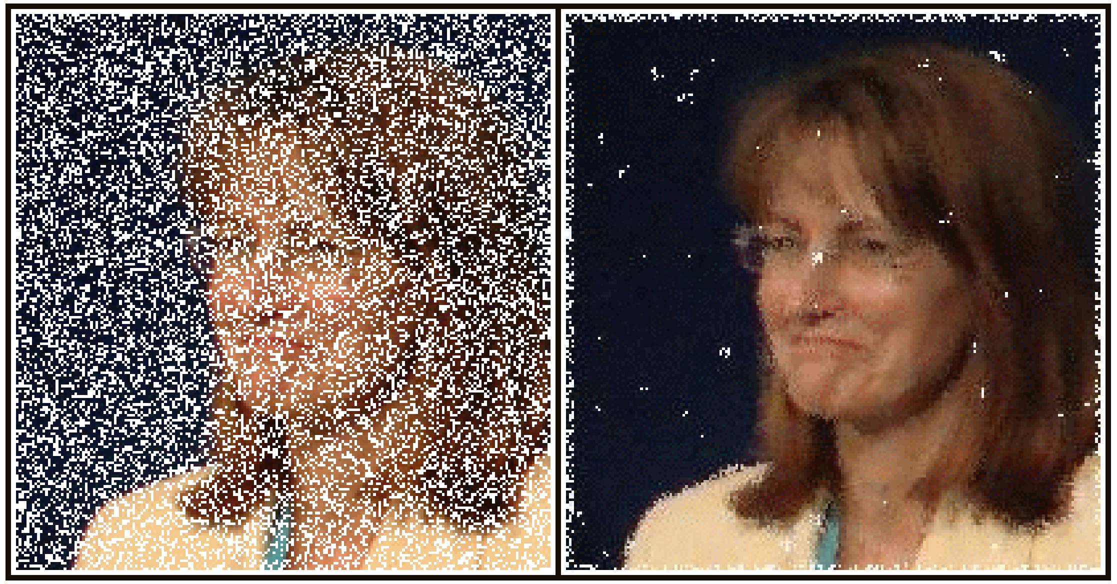
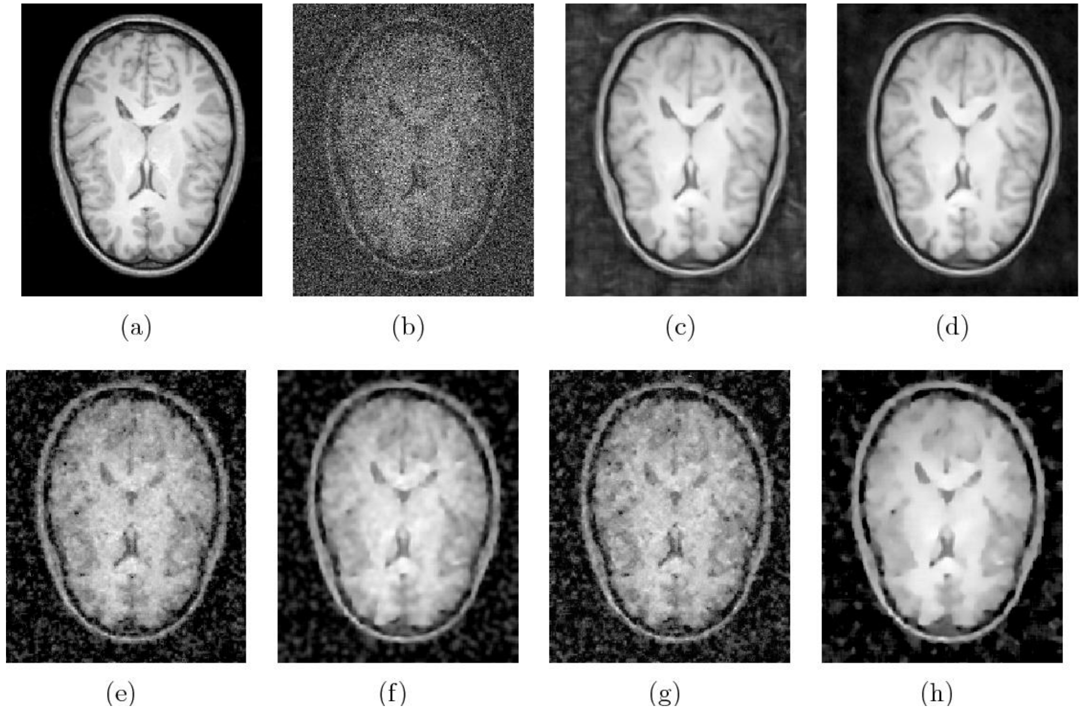


Image restoration



Medicine
(MRI denoising)

contents

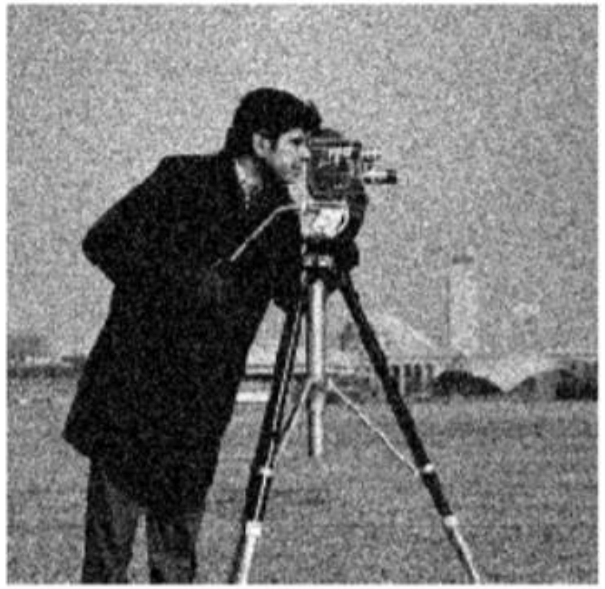
- **Noise and filtering**
- Frequency representation
- Decimation
- Interpolation

Gaussian Noise

- Gaussian noise is an additive noise that can appear in images due to the system electrical circuitry.
- This noise is independent of signal strength and independent at each pixel (IID- independent and identically distributed).

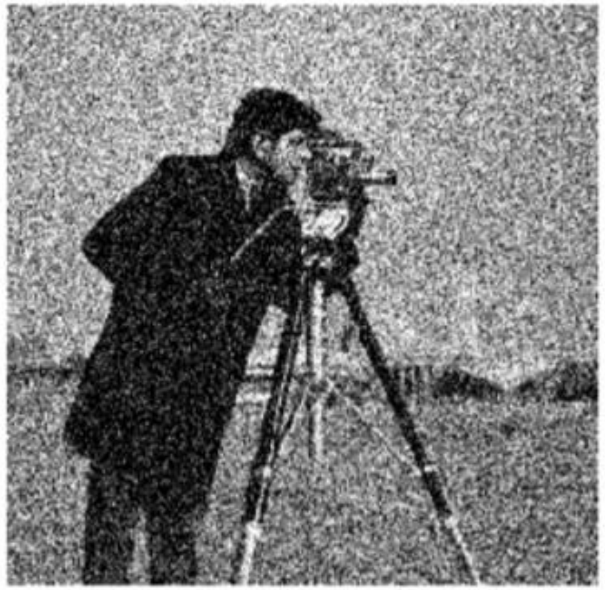
$$p(z) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(z-\mu)^2}{2\sigma^2}}$$

Gaussian Noise σ=0.01



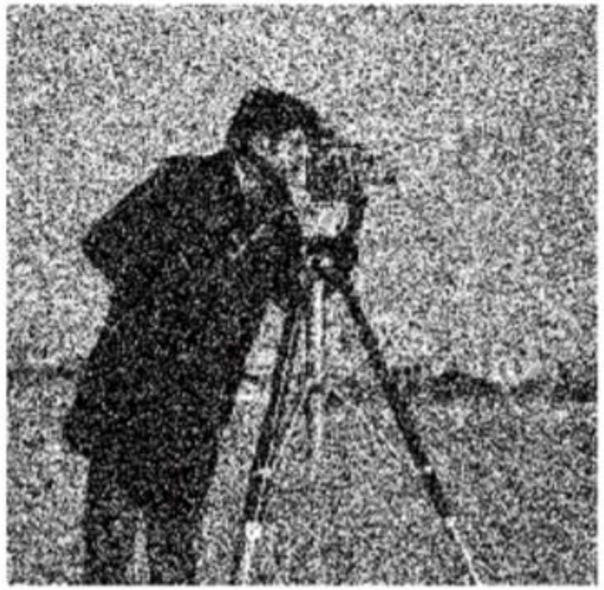
SNR=34.0206

Gaussian Noise σ=0.05



SNR=19.1825

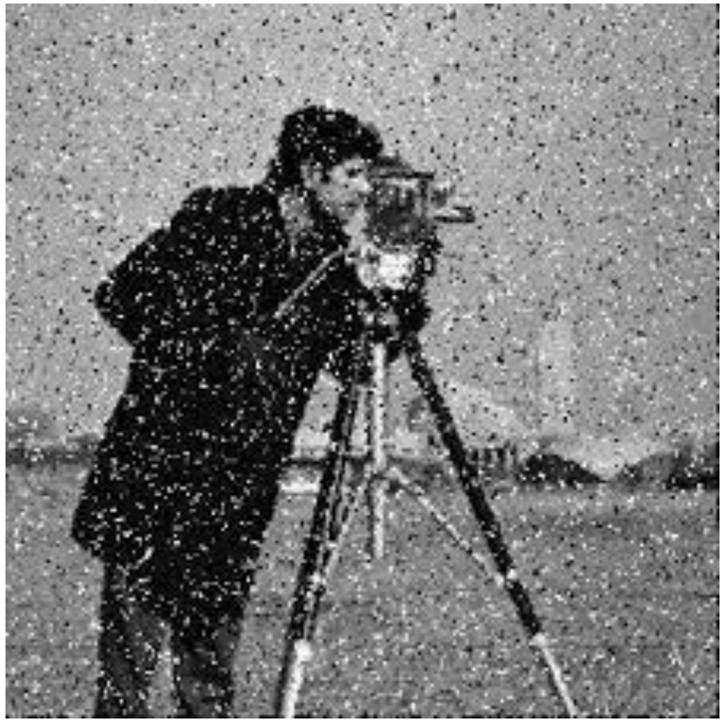
Gaussian Noise σ=0.1



SNR=13.7121

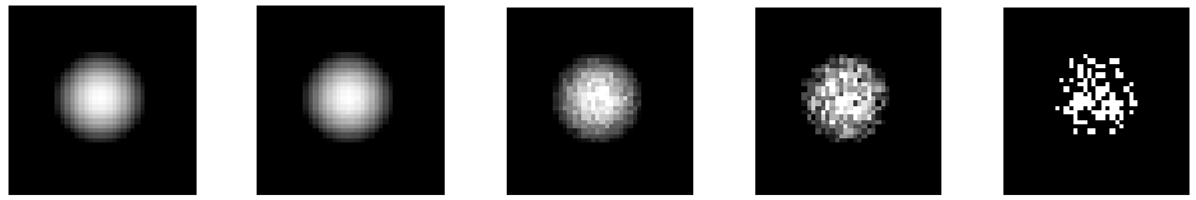
Salt & Pepper noise

- Noise that can be caused by analog-to-digital converter errors, bit errors in transmission, etc.
- This noise is **not** additive to the signal strength (a replacement of original value with noise value).
- This noise is independent of signal strength and independent at each pixel.

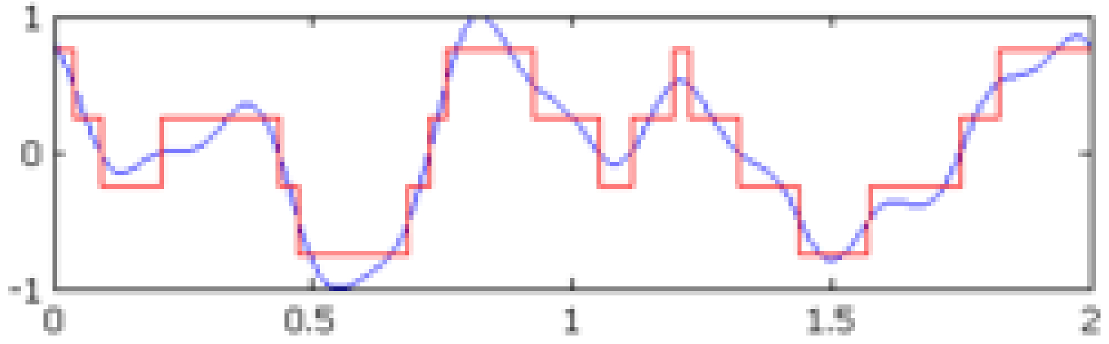


And some more noise

- **Shot noise** - caused by statistical quantum fluctuations, that is, variation in the number of photons sensed at a given exposure level in the darker parts of an image (where there are just few photons that enter each pixel “bin”). Modeled as Poisson noise.



- **Quantization noise** – caused by quantizing the pixels of a sensed image to several discrete levels (analog to digital conversion).



Noise reduction with LTI filters

- **LTI (LSI) filters** are also known as **convolution filters** or **kernel filters** and are a known solution for the noise problem.
- They are **linear** operators, which involve weighted combinations of pixels in small neighborhood, The combination is determined by the filter’s *kernel*.
- The same kernel is **shifted** to all pixel locations so that all pixels use the same linear combination of their neighbors.
- Shifting the output after the weighted combination vs. shifting the input and then doing weighted combination is the same- so **shift-invariance**.
- That’s why it’s called **linear shift-invariance** filter.
  - LSI for short, but more commonly known by the name of the 1D signal filter- LTI, linear time-invariant.

Convolution

- Works on LTI filters.
- Let f be the image, h be the kernel of size $(2k+1)X(2k+1)$ (k is a chosen integer), and g be the output image:

$$g(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] f[i-u, j-v]$$

  – **Note:** by definition, this operator flips the kernel both horizontally and vertically.
- This operation is called **convolution operator** and is more compactly notated as:

$$g = h * f$$

- Very similar to **cross correlation** only here the flip of the kernel is done (also here the center of the template is considered- in CC the upper left usually).

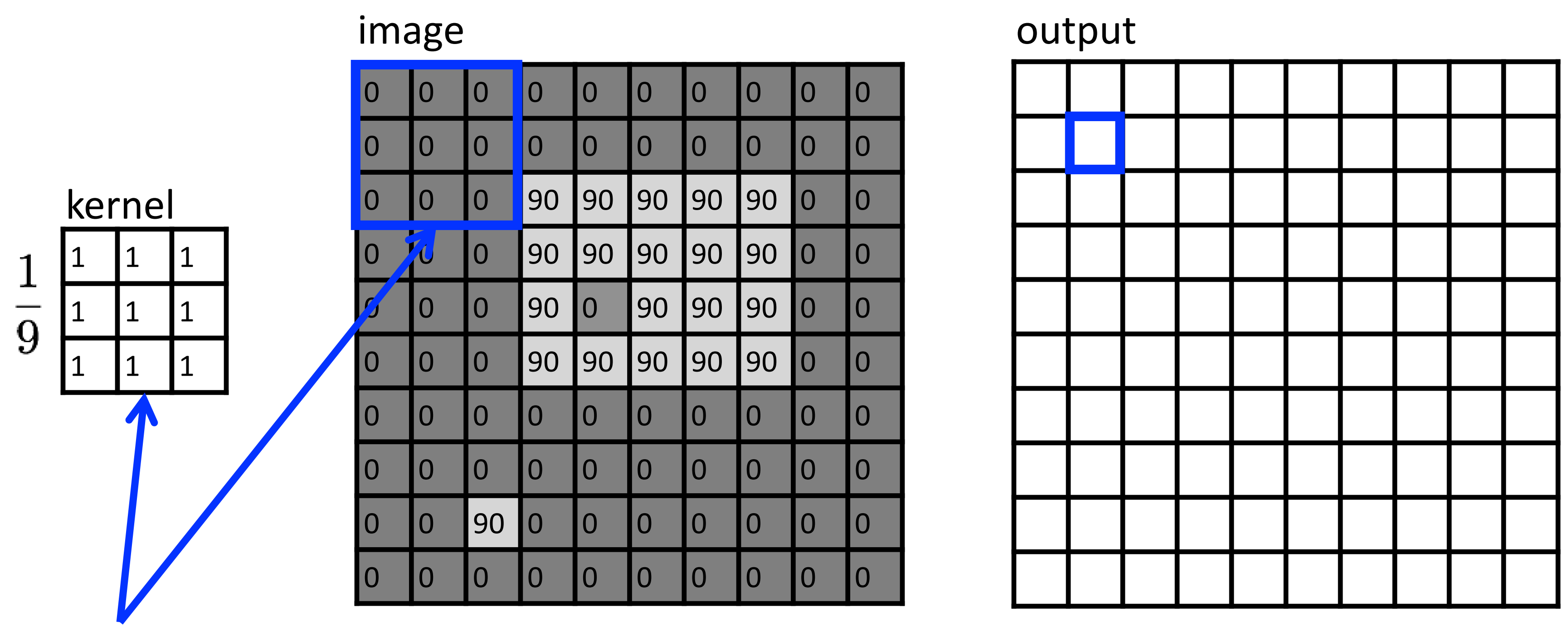
Example: mean filter

- The kernel is:

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

- Replaces pixel with local average.
- Has smoothing (blurring) effect.
- The kernel can be in any other size as well (see .ipynb).

Run the filter

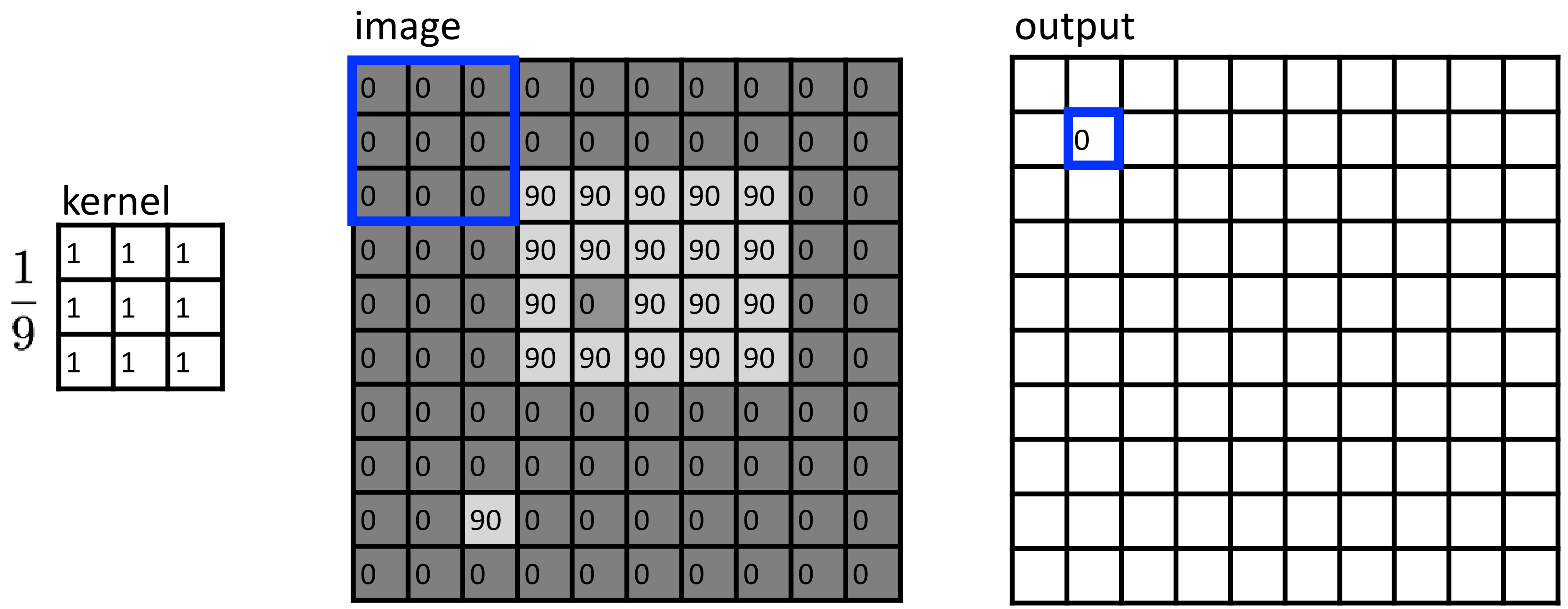


$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Note that we assume that the kernel coordinates are centered.
Here the kernel is symmetric horizontally and vertically, so the flipping is not noticeable.

Run the filter



kernel

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

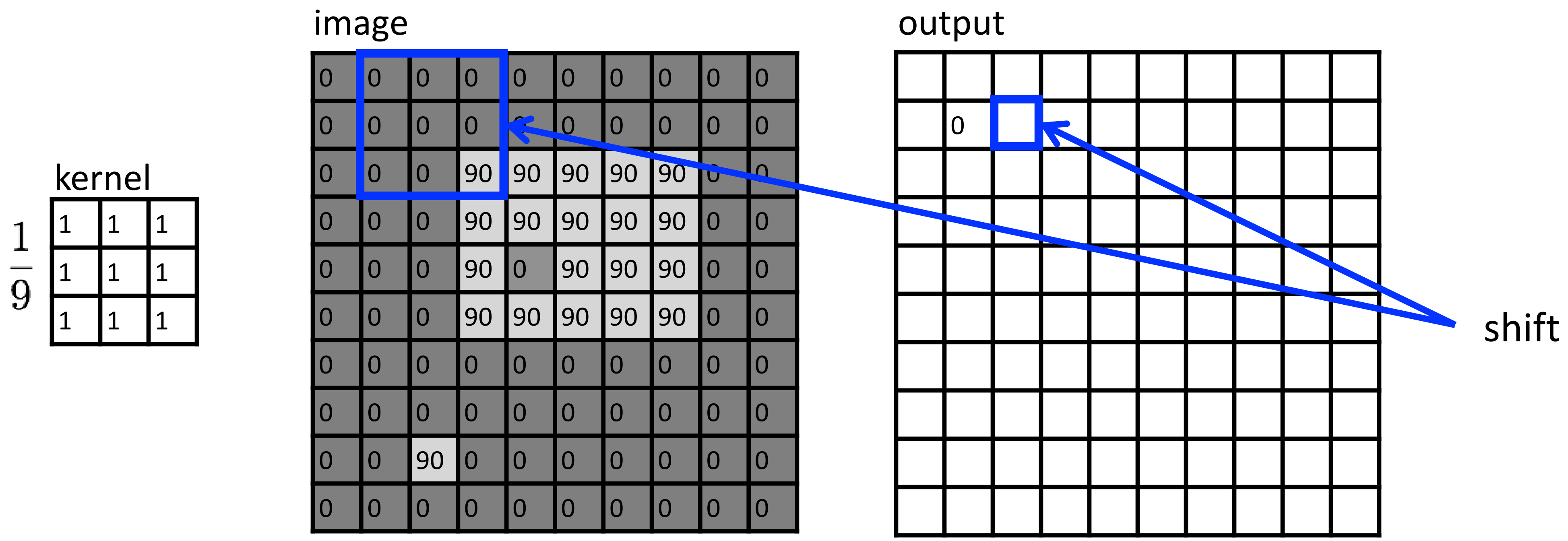
image

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output

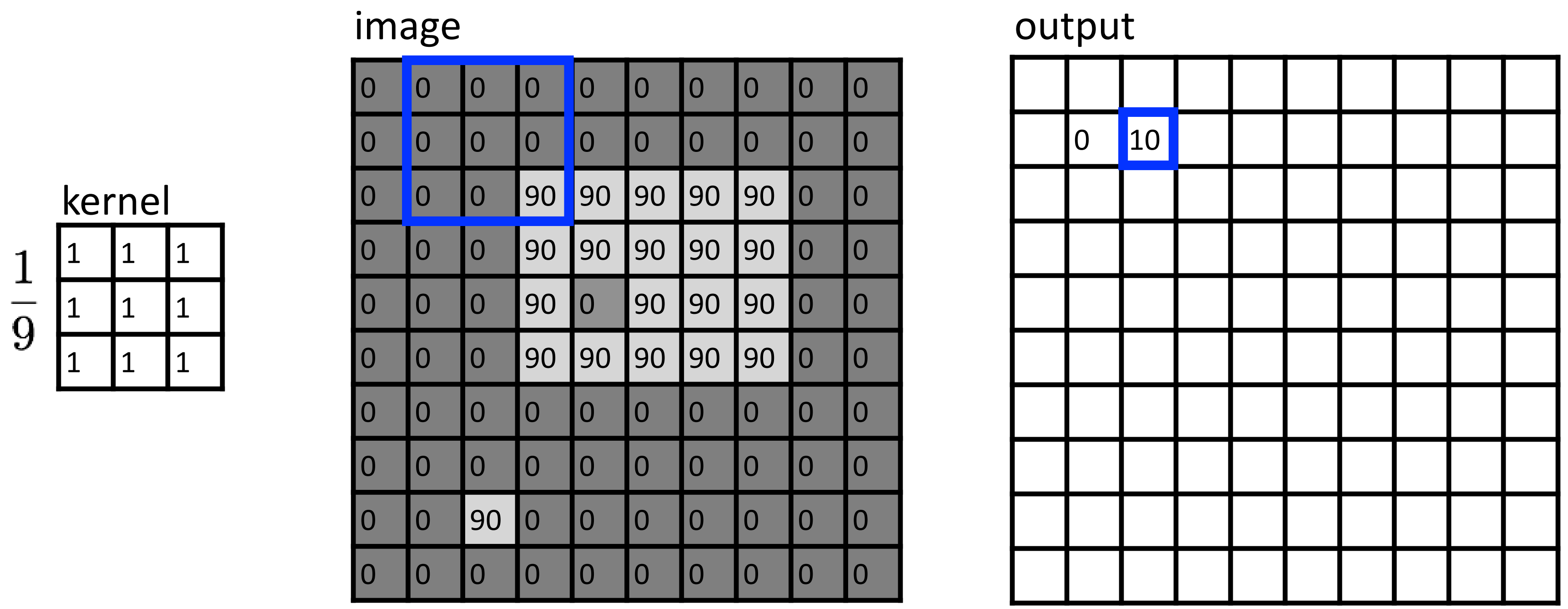
	0								

Run the filter



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Run the filter



kernel

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

image

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output

	0	10							

Run the filter

kernel

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

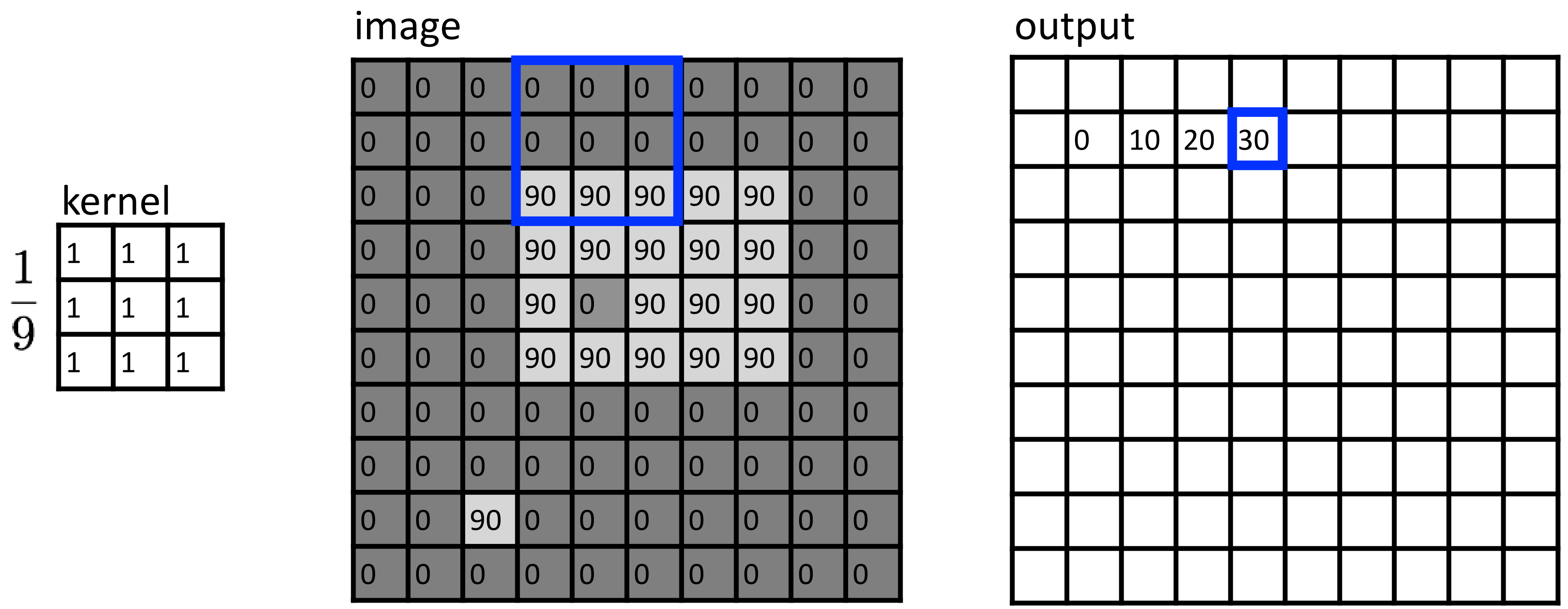
image

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output

	0	10	20						

Run the filter



kernel: $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

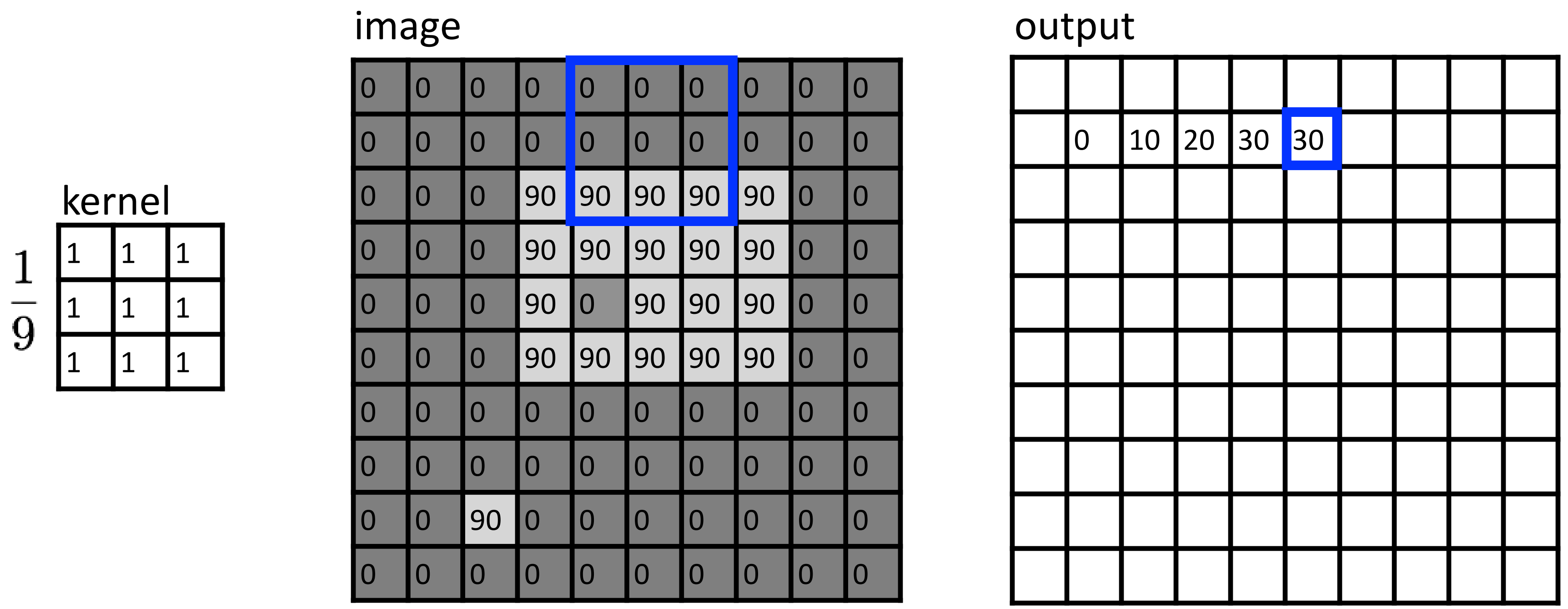
image

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output

	0	10	20	30					

Run the filter



kernel

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

image

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output

	0	10	20	30	30				

Run the filter

kernel

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output

	0	10	20	30	30	30			

… and the result is

kernel

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

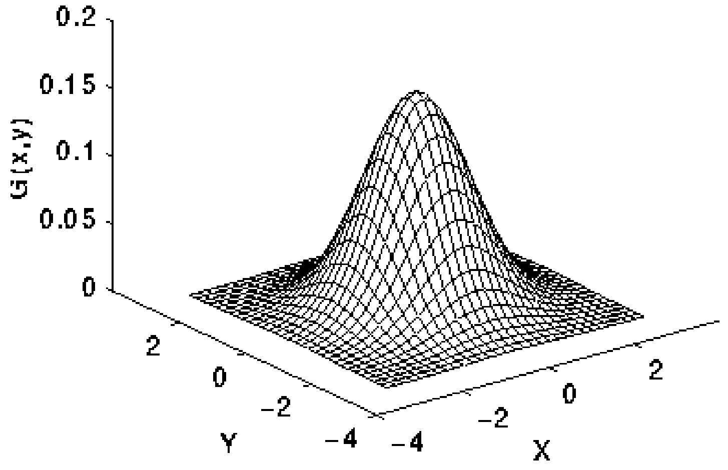
image

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	0	10	20	30	30	30	20	10	
	10	10	10	10	0	0	0	0	
	10	10	10	10	0	0	0	0	

Gaussian filter



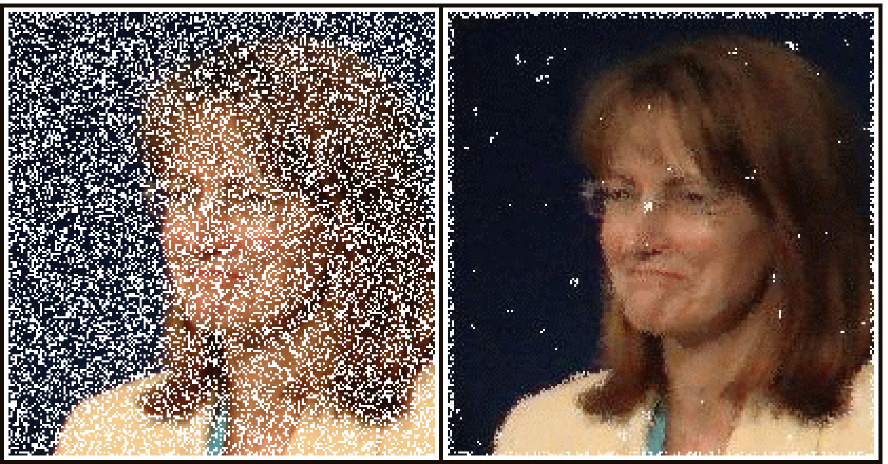
- Another kind of blur filter.
- this filter can be controlled by its size and STD.

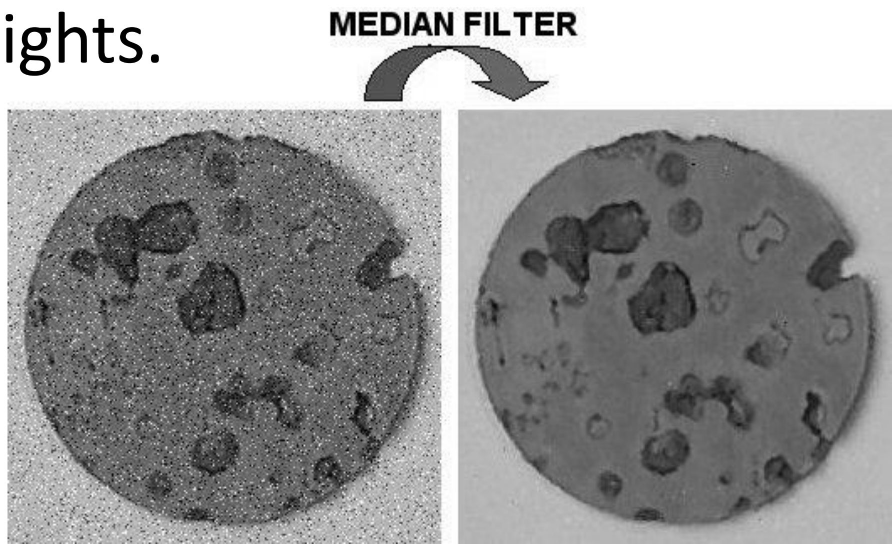
$$h(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- The kernel is discretized to bins according to the wanted kernel size.
- Isn't Gaussian function infinite?
  - Most often, the kernel cuts out the remaining lower bins (usually at 2-3 σ).
- **Both Gaussian and mean filters are good against Gaussian noise, but not effective against S&P noise.**

Median filter

- Takes the median value from the given neighbors.
- For example:
  - The median of [1, 0, 100] is 1.
- **Median filter is good against salt and pepper noise & against Gaussian noise (but not as effective).**
  - Can be considered as a blur filter, but also has edge preserving properties.
- Median filter is more computationally expansive than mean/gaussian.
- This filter is not LTI because it’s not linear on its weights.





contents

- Noise and filtering
- **Frequency representation**
- Decimation
- Interpolation

Fourier transform

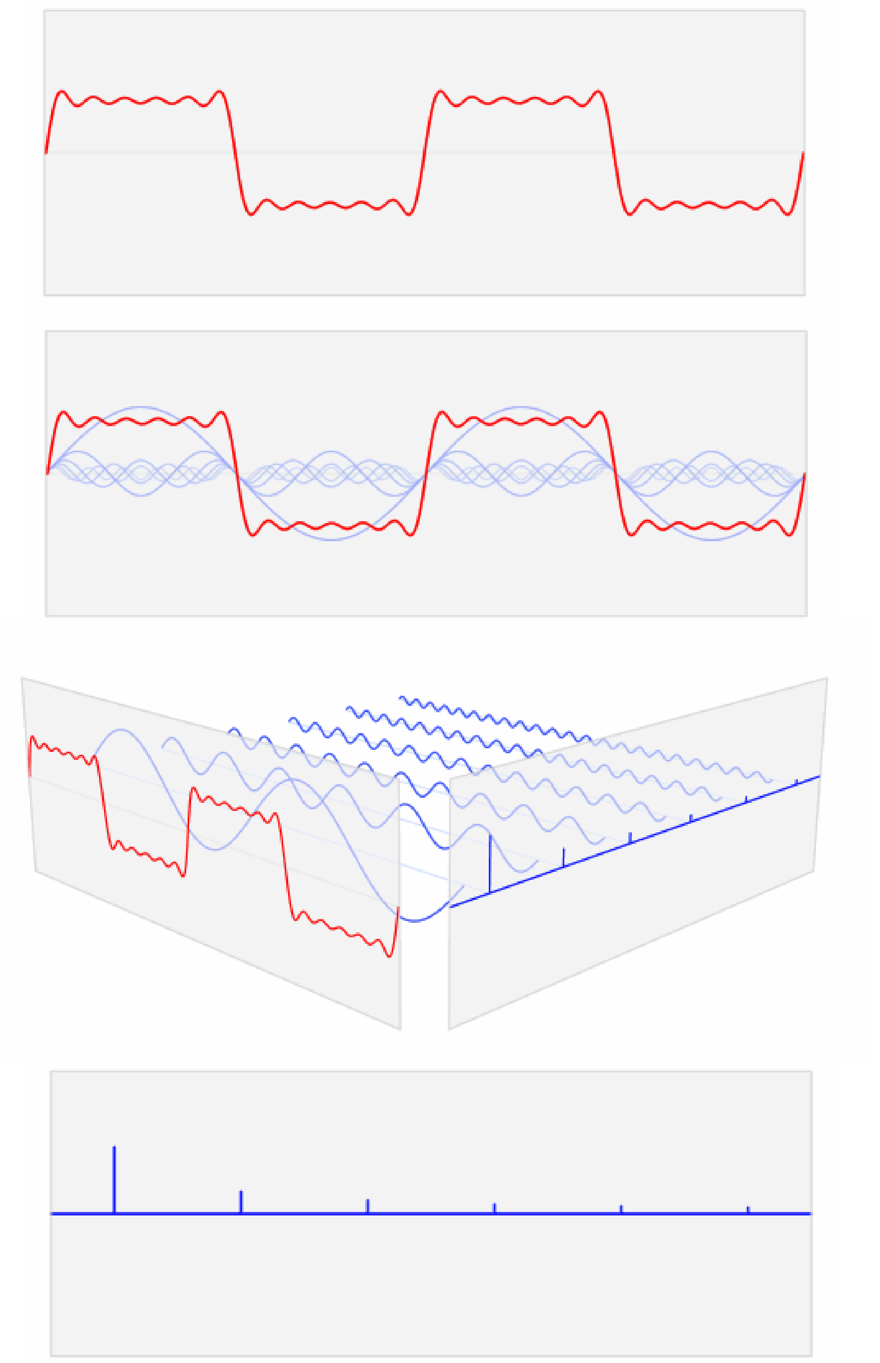
- Each periodic signal can be represented as a collection of cosine functions added together- these is called **Fourier transform (FT)**.
  - [we will leave the exact definition aside… This is not a signal processing course]
- A cosine function can be represented by 3 variables:
  - Frequency: f [Hz]
  - Amplitude: A
  - Phase: ϕ

$$g(t) = A \cdot cos(2\pi f t + \phi)$$

- Demo: https://www.desmos.com/calculator/metjpkf2e5

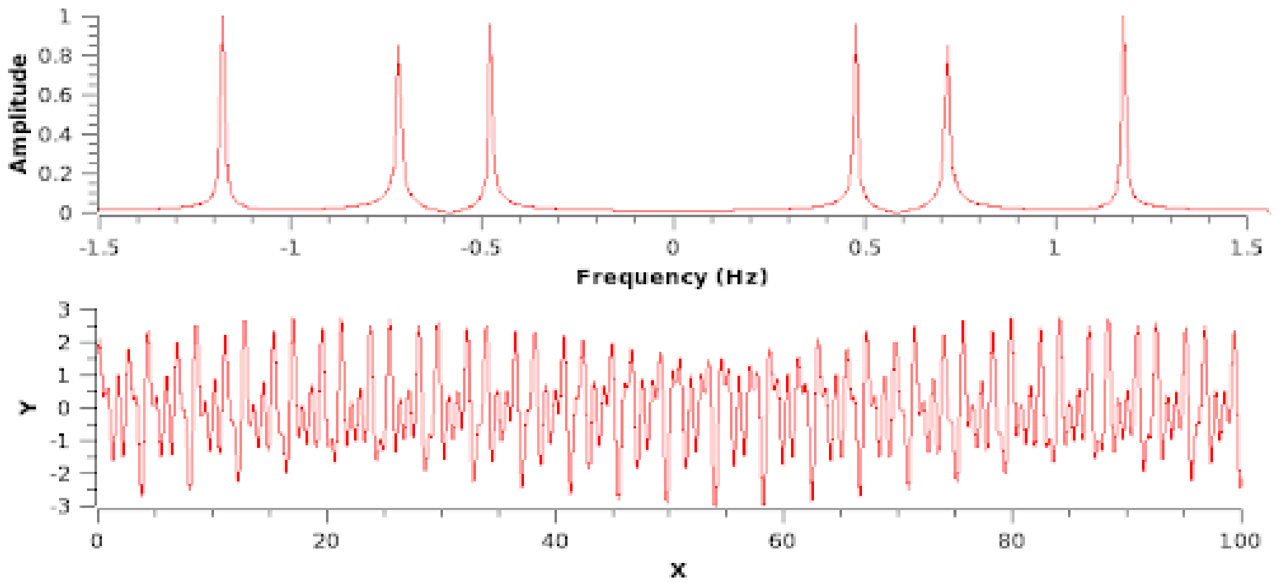
Fourier transform

- FFT (fast Fourier transform) is an efficient algorithm to decompose a signal into its collection of added cosine functions: frequency, amplitude and phase.
- Most of the times, when talking about the FFT of a signal, you'll see graph of the frequencies and their belonging amplitudes (the phase is omitted).
- FFT examples: http://www.jezzamon.com/fourier/



Fourier transform

- When doing FT/FFT of a signal, one also get negative frequencies.
  - Why? Again, out of scope… this is the uniqueness of the Fourier transform and its ability to work on complex signals.
  - When the signal is real, the Fourier transform is symmetric and hence most of the time the negative part is erased.
  - The same is true for 2D signals, but there we tend to keep the symmetric part.

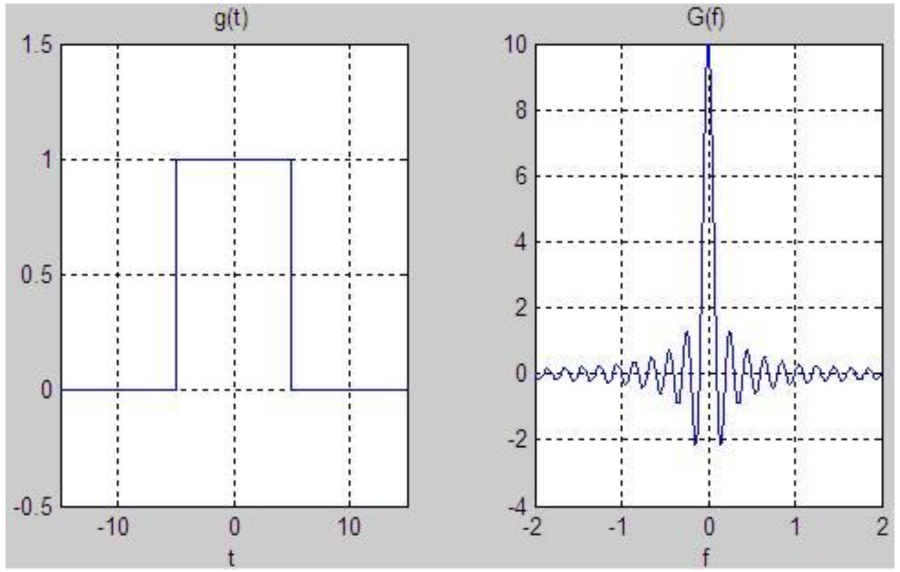


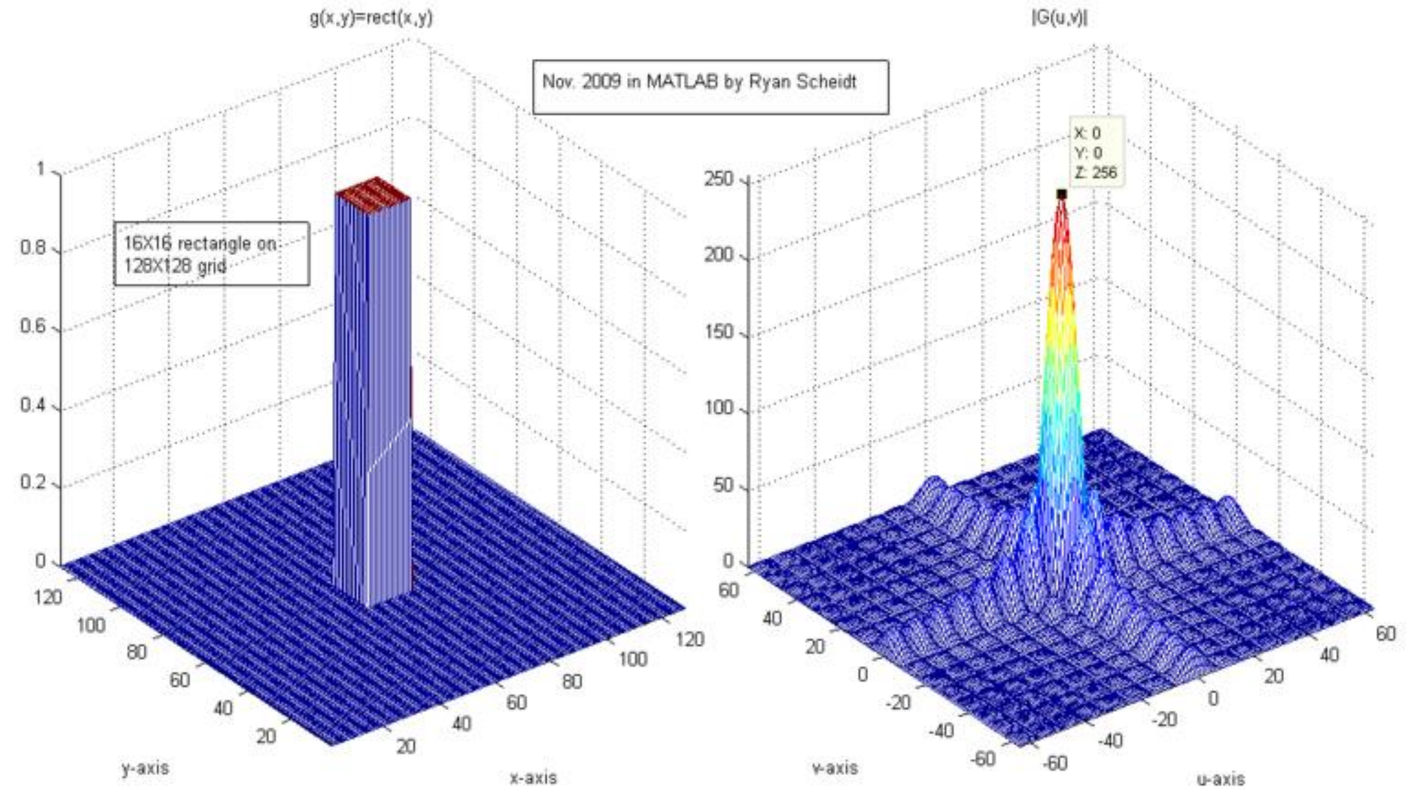
2D FFT

- Like in 1D signals, 2D signals also have FFT.
- 2D cosine wave is represented as such:

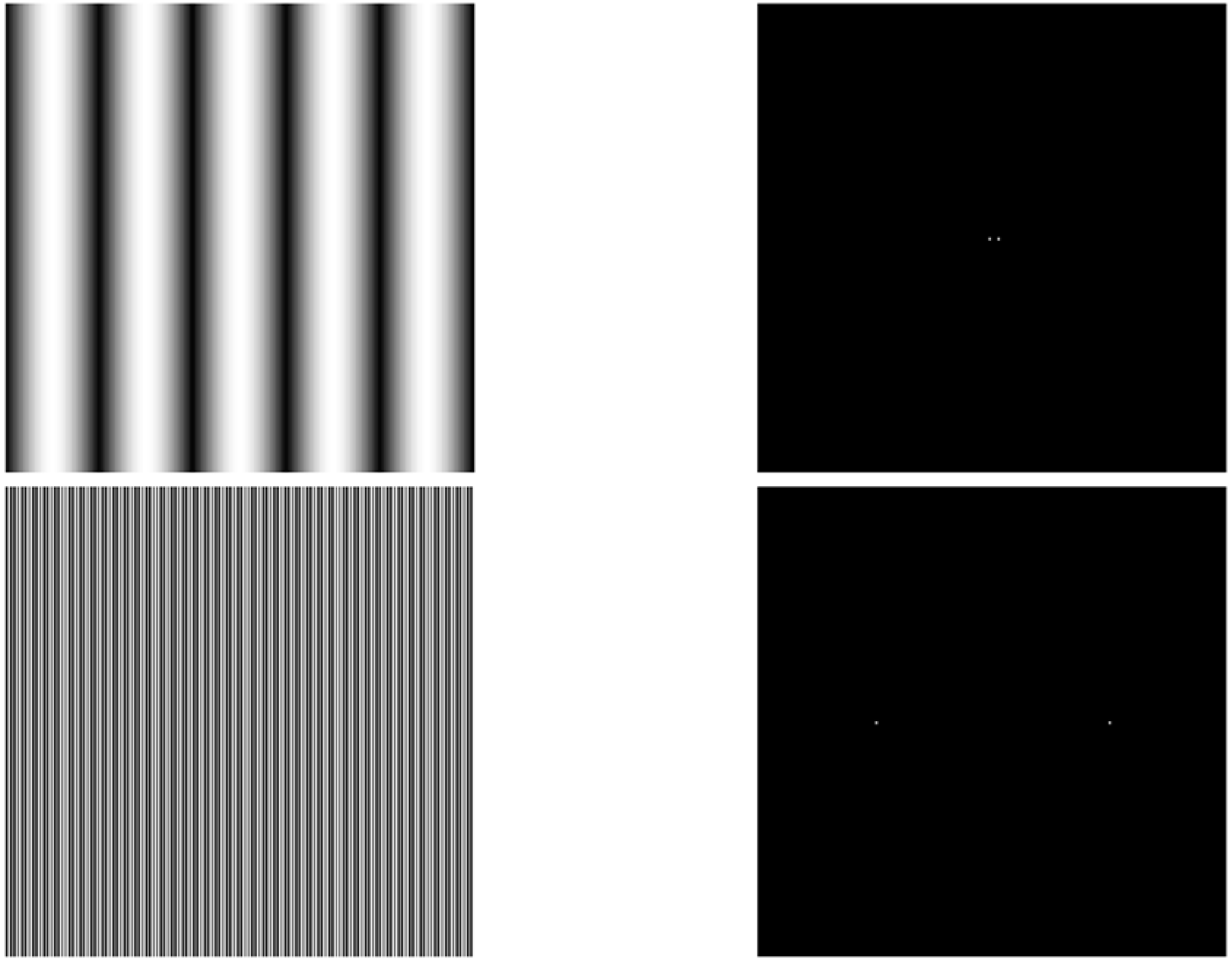
$$g(x,y) = A \cdot cos(2\pi(ux + vy) + \phi)$$

- The FFT of such a signal returns the amplitude, phase and directional frequency (u & v)

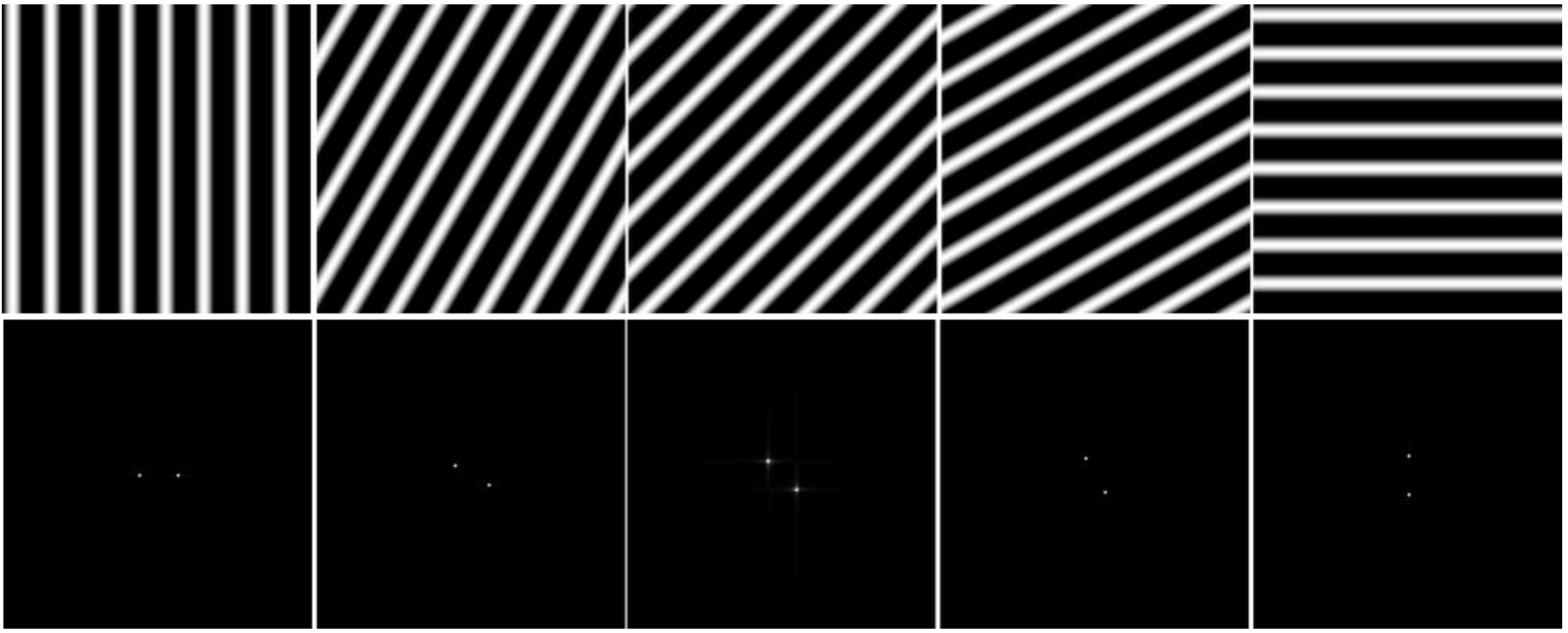




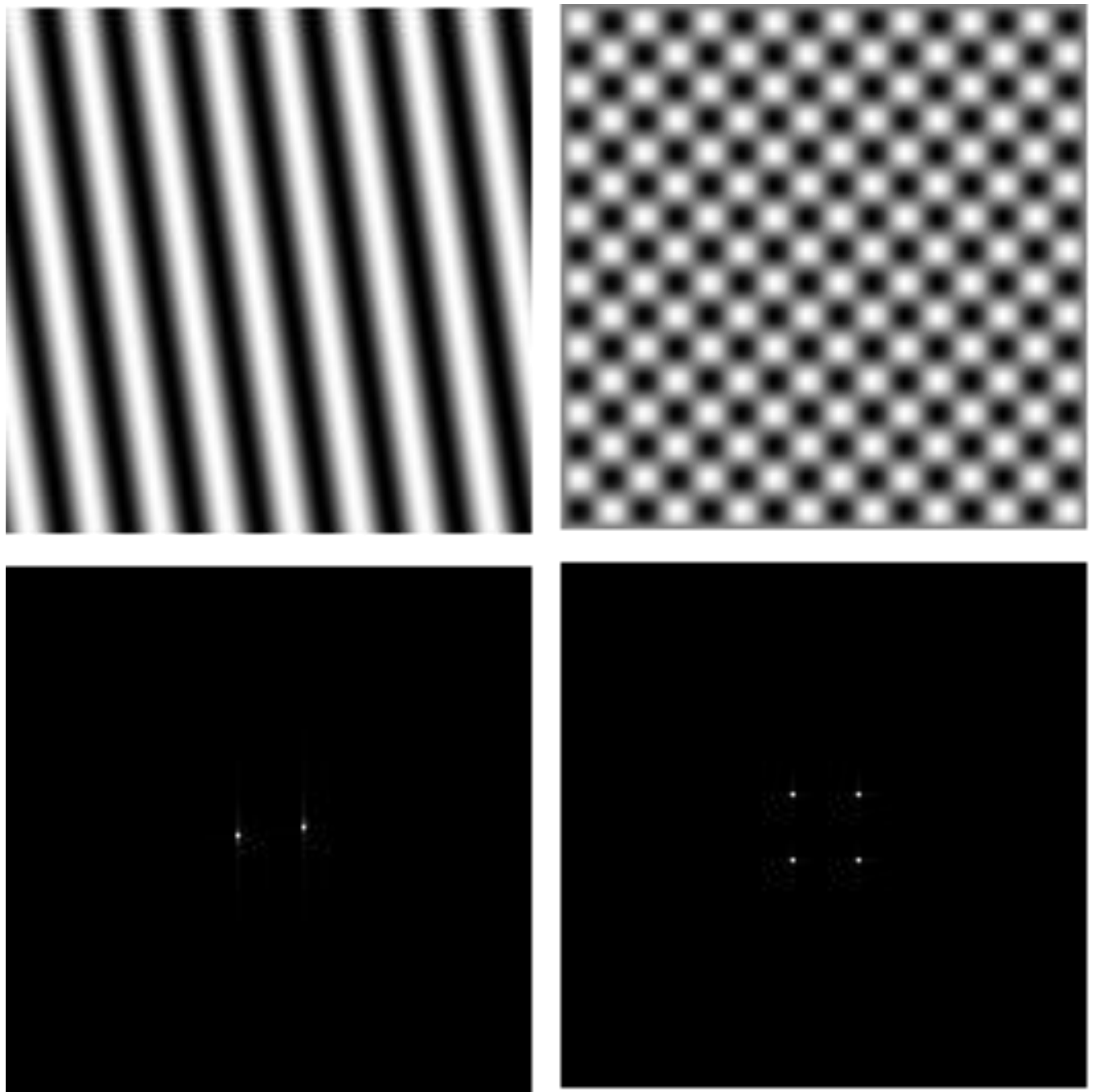
2D FFT examples



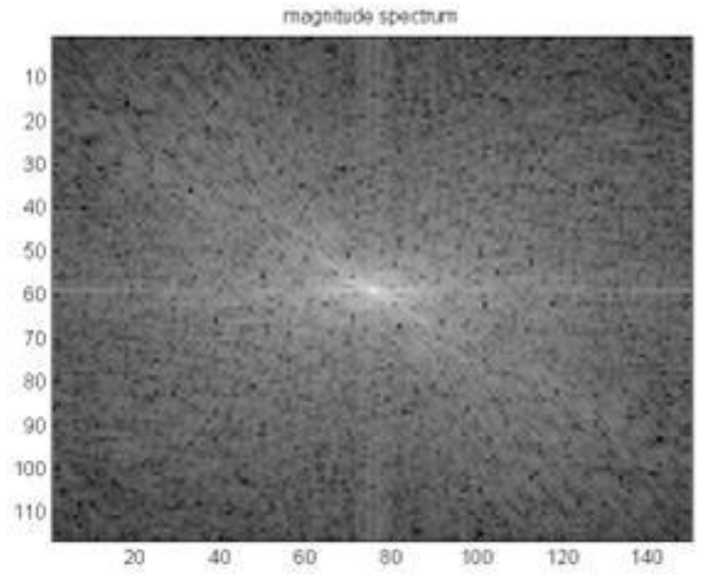
2D FFT examples

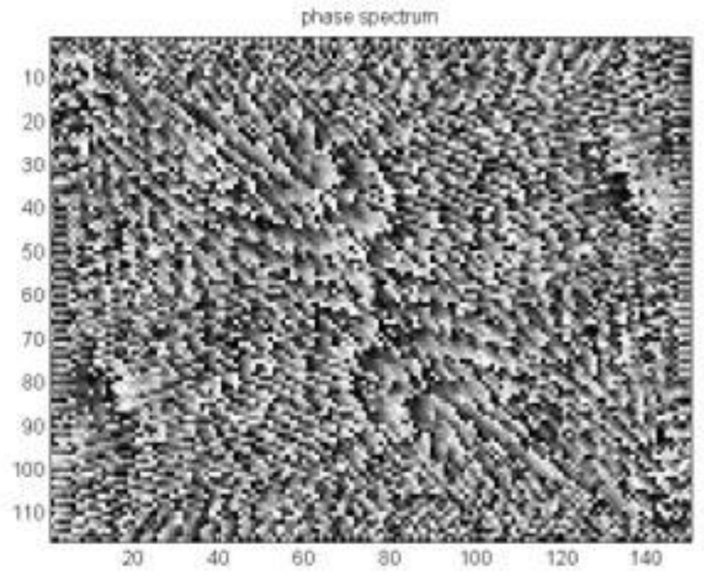


2D FFT examples



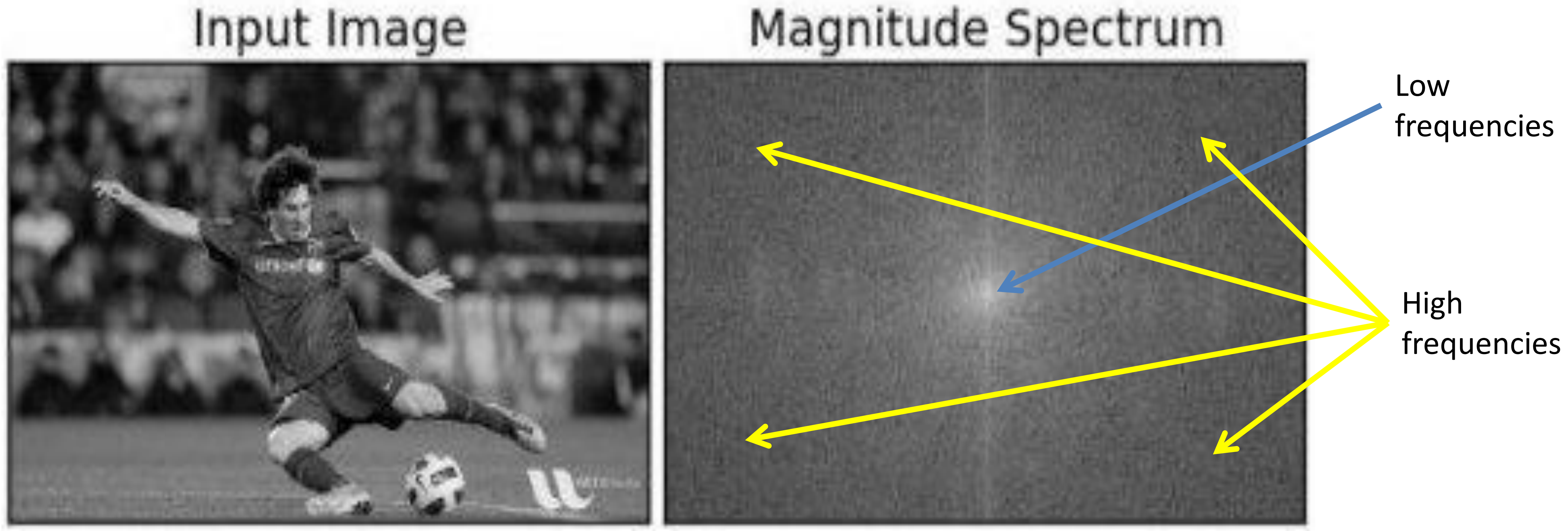
2D FFT examples





2D FFT

- In audio signals (or any other 1D signals) Lower frequencies change less over time than higher frequencies.
  - In images, the change is represented in change in distance, so images that changes slowly from pixel to pixel has more lower frequencies then others.
- Natural images are mainly built from low frequencies.



2D FFT

- 2D FFT demo: http://www.jezzamon.com/fourierl#jpegs
  - [Actually DCT (discrete cosine transform) but it's a good demo none the less)

Convolution in frequency domain

- Recall: in time (space) domain:

$$g(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] f[i-u, j-v]$$

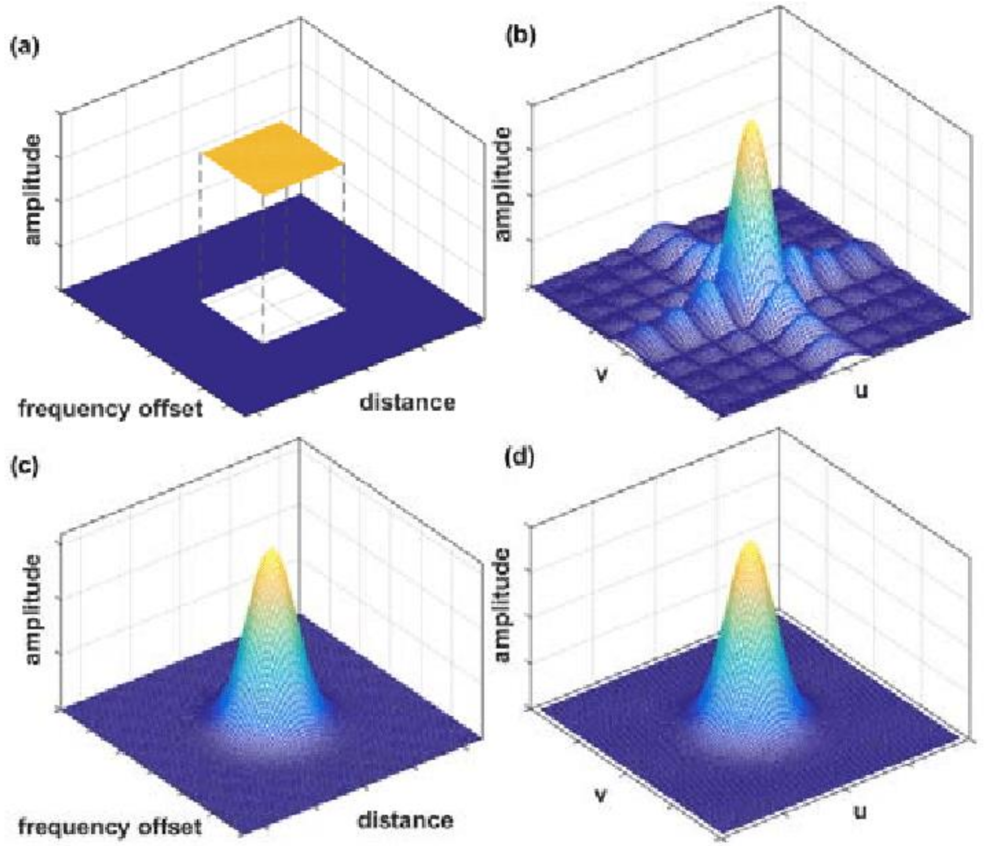
$$g = h * f$$

- In frequency domain- simple multiplication:

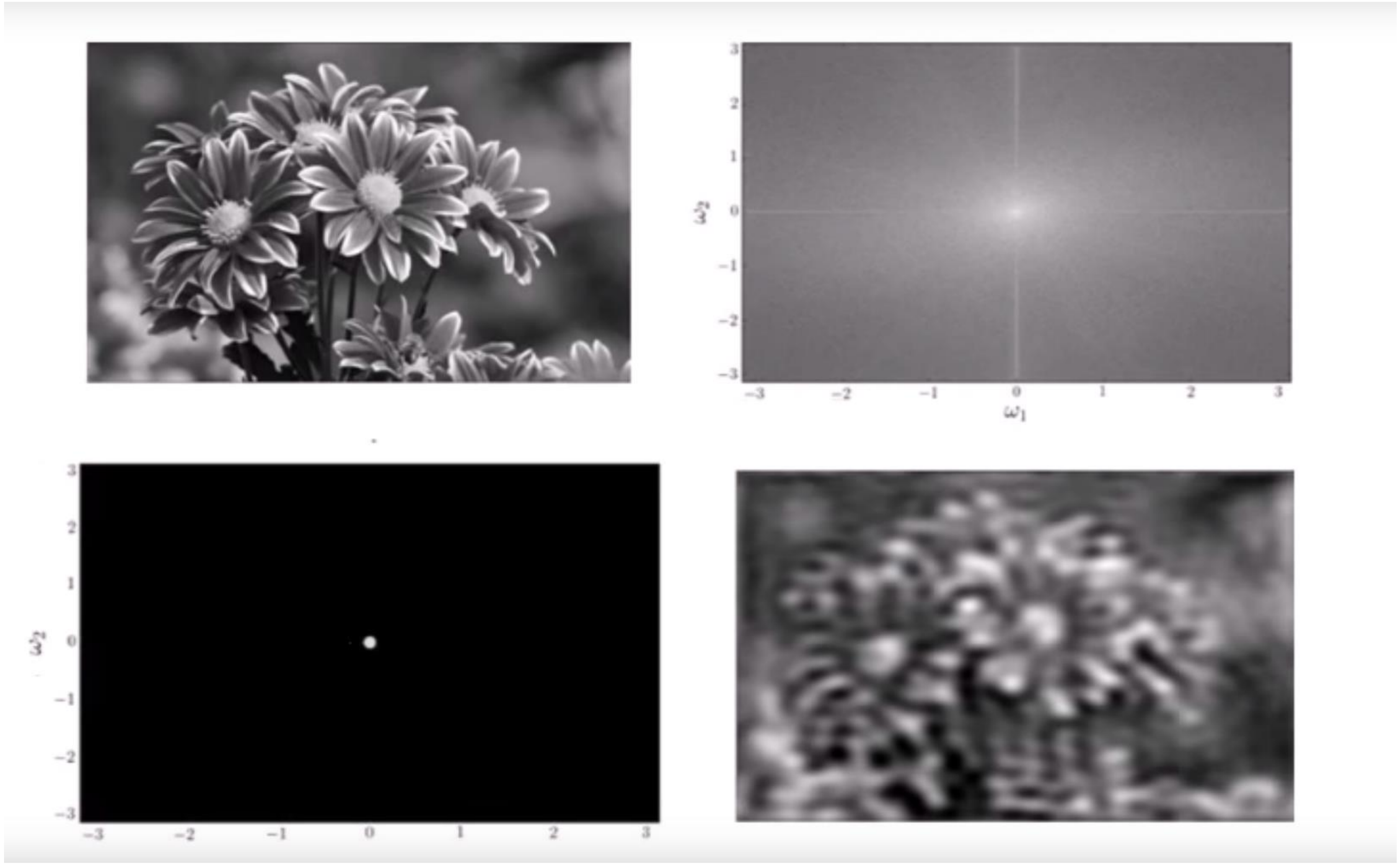
$$G = H \cdot F$$

Low-pass filters

- Both mean and Gaussian filters are considered low-pass filters because in the frequency domain, they have higher values in the lower frequencies- and when multiplied with frequency spectrums, the high frequencies get smaller.
- When image is left only with the lower frequencies, the rapidly changes parts of the image (e.g.: edges, noise) are smoothen.

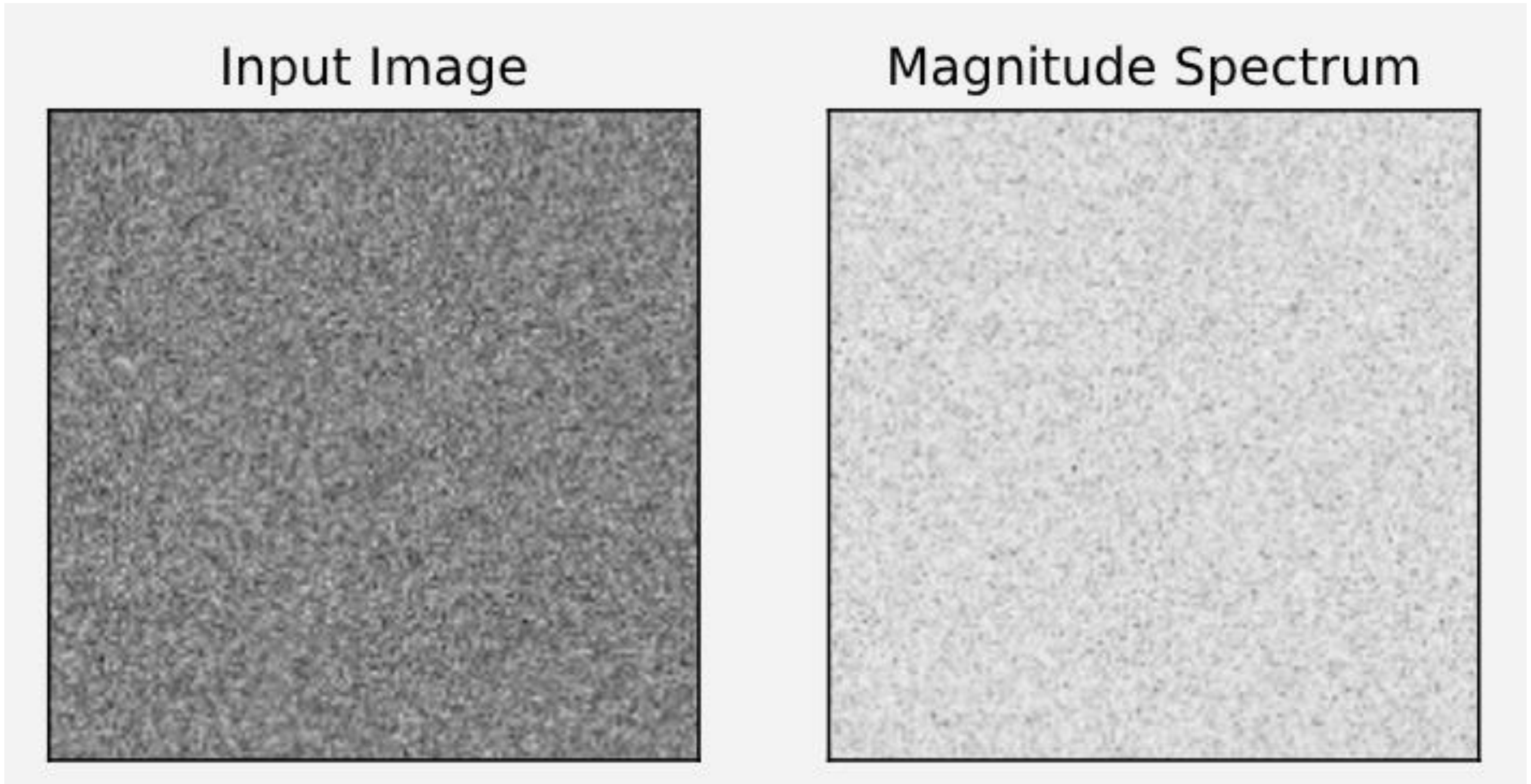


LP example



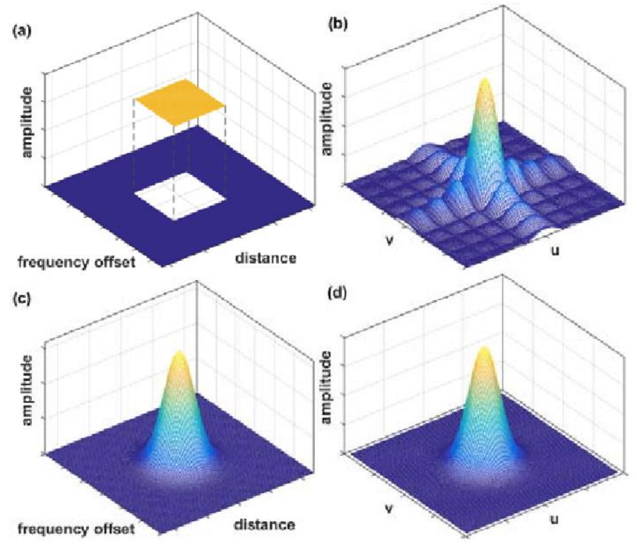
FFT of gaussian noise

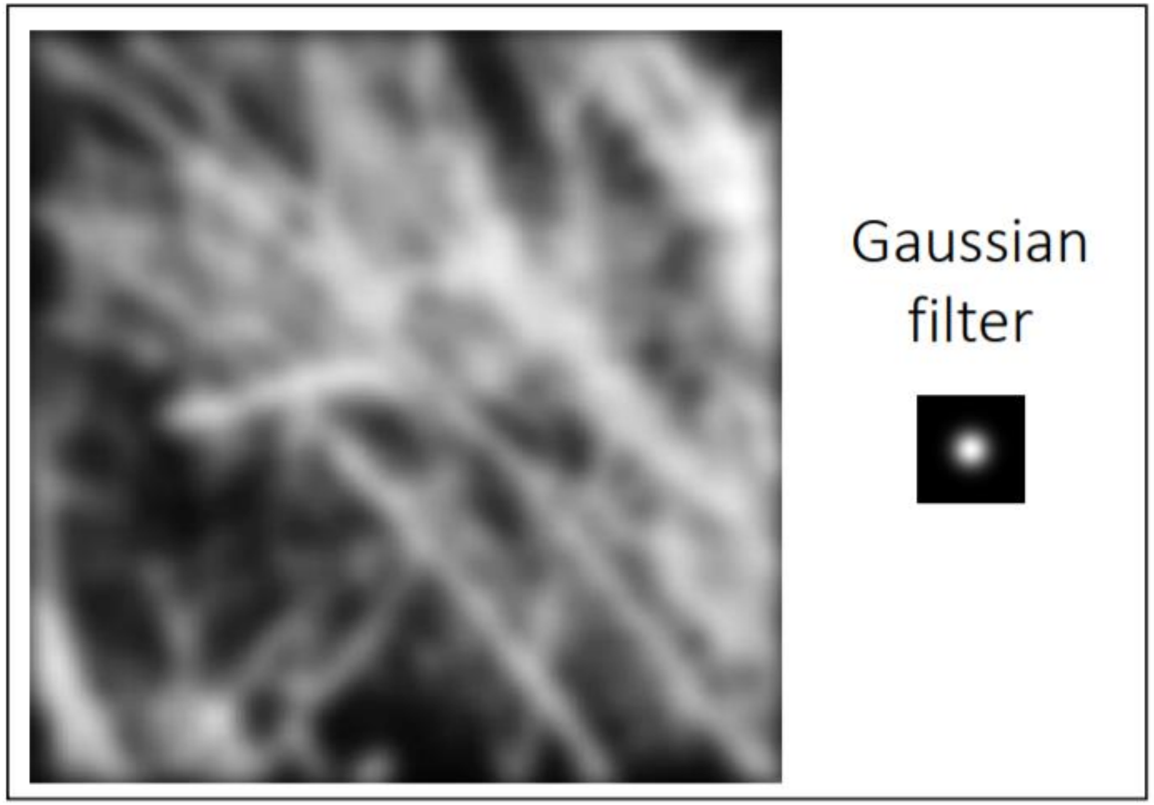
- Since gaussian noise (AWGN) is distributed along all frequencies, LP filter reduce this kind of noise significantly.

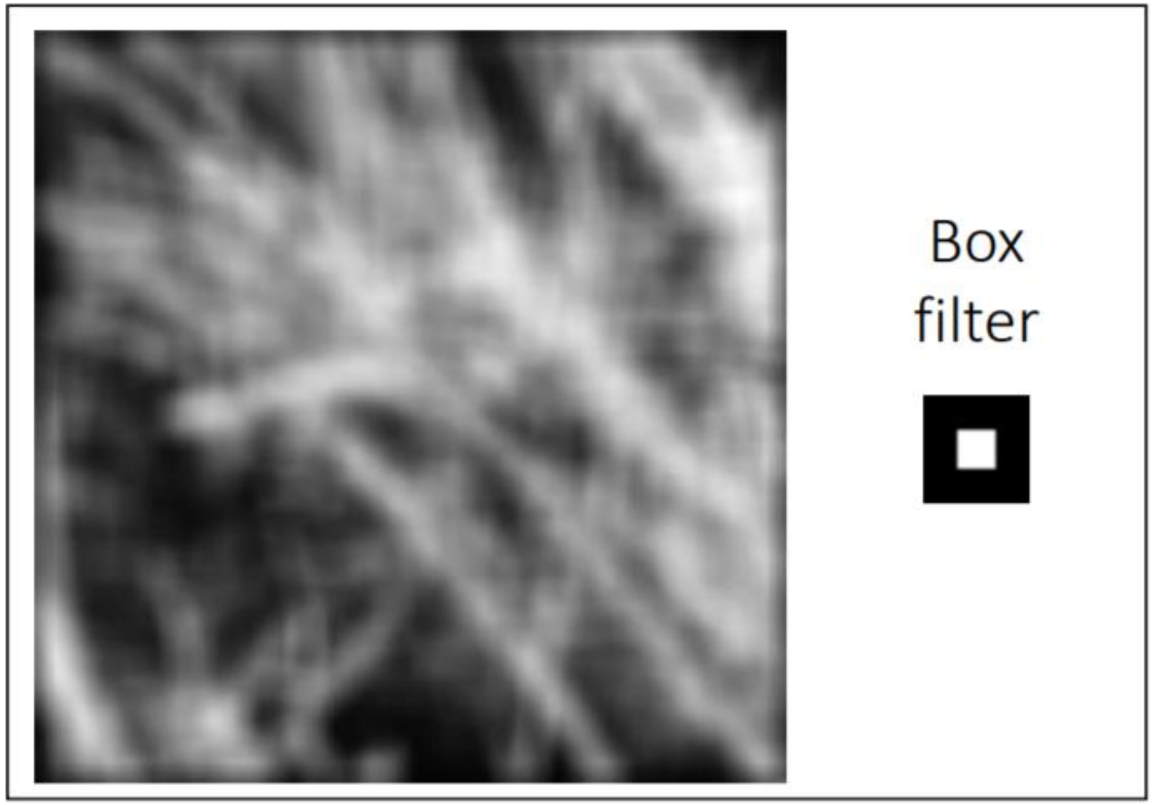


Mean vs. Gaussian filter

- Since Mean filter has some high values in high frequencies, edge artifacts sometimes remains.
- Gaussian filter has less artifacts in higher frueqancies.

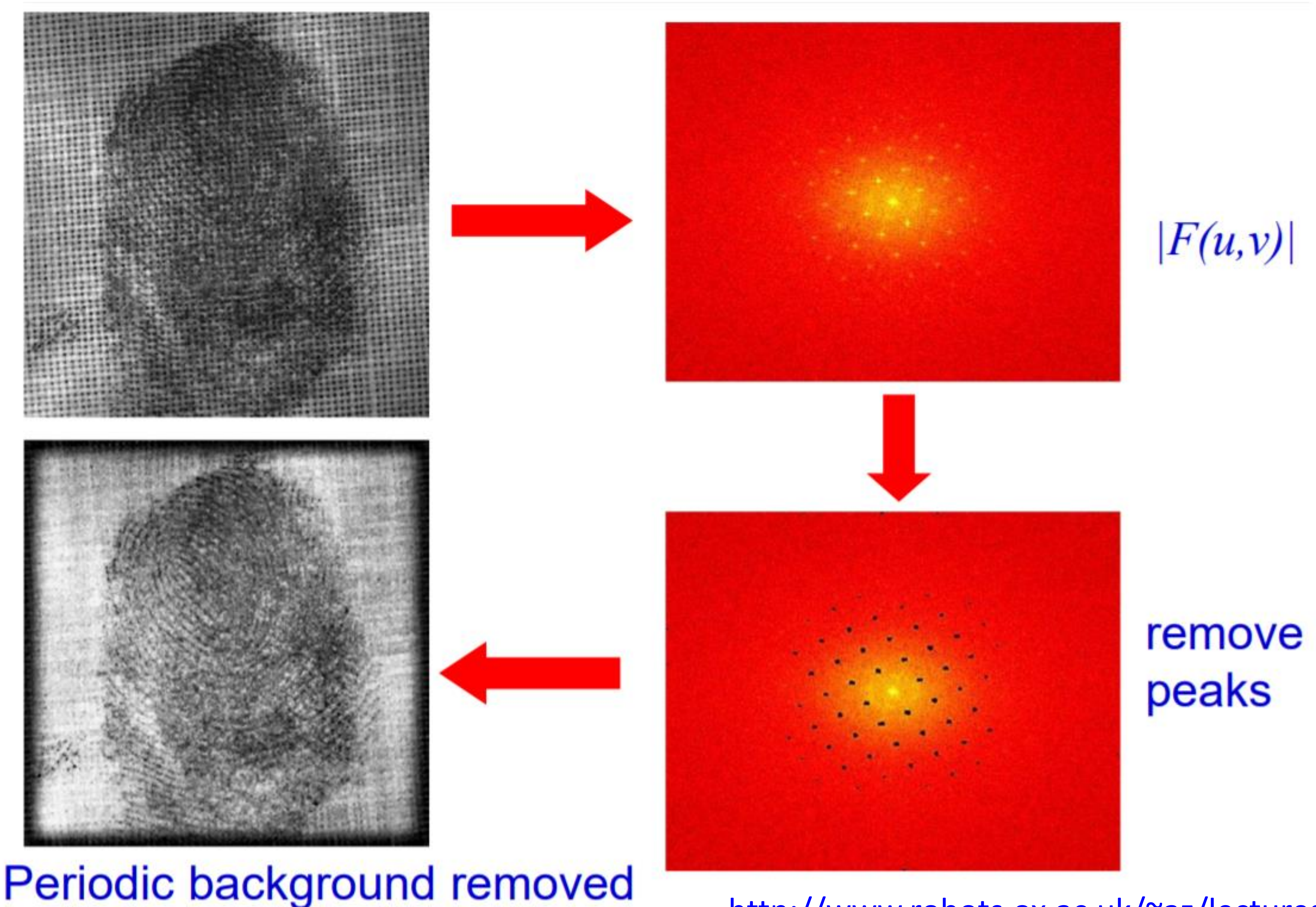






More applications with frequencies

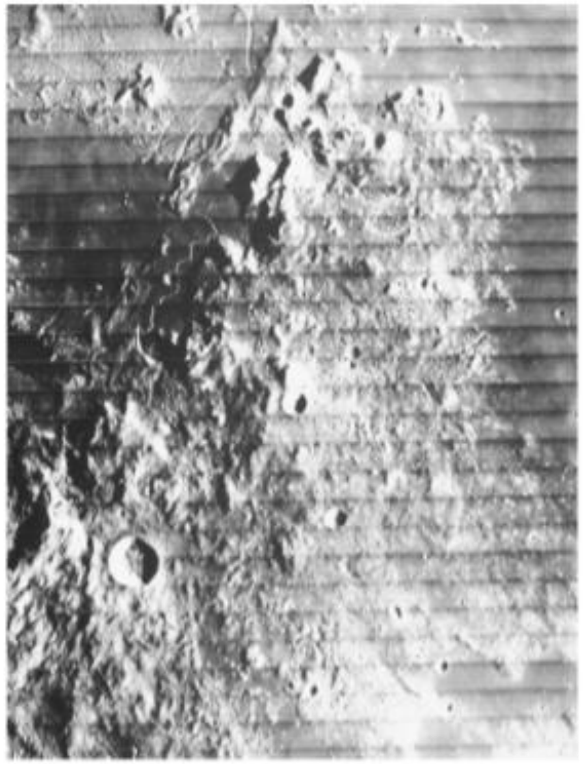
- Forensics

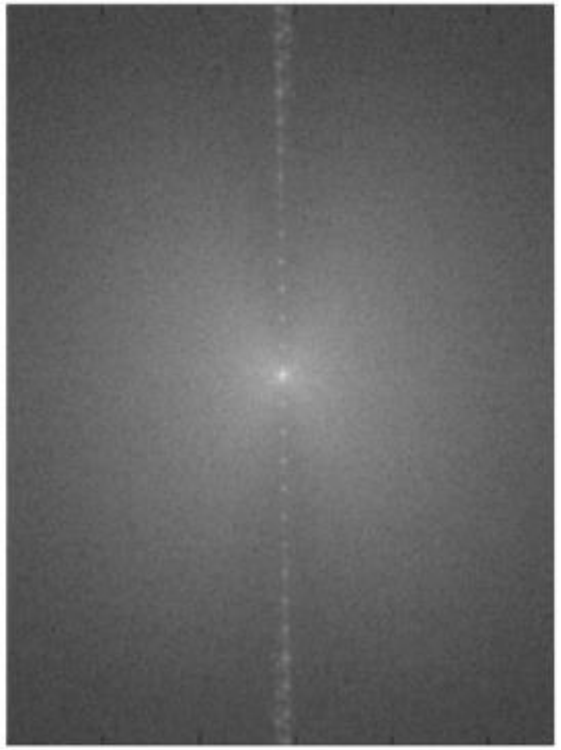


http://www.robots.ox.ac.uk/~az/lectures/ia/lect2.pdf

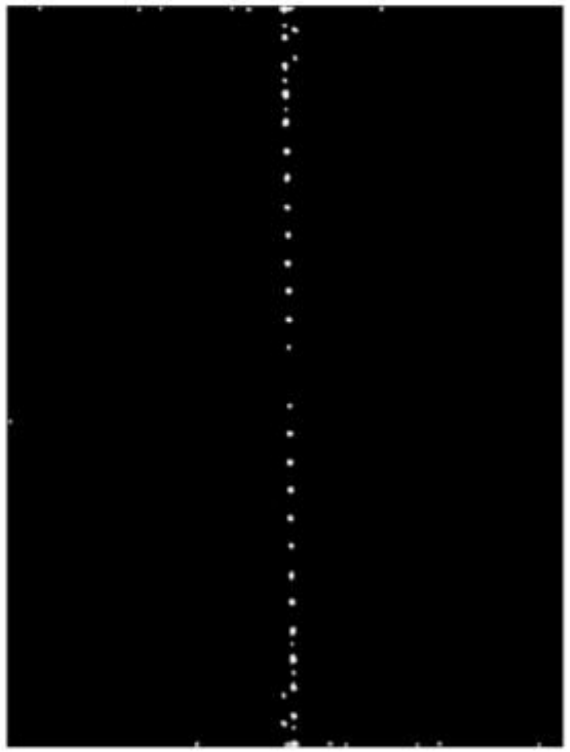
More applications with frequencies

Lunar orbital image (1966)

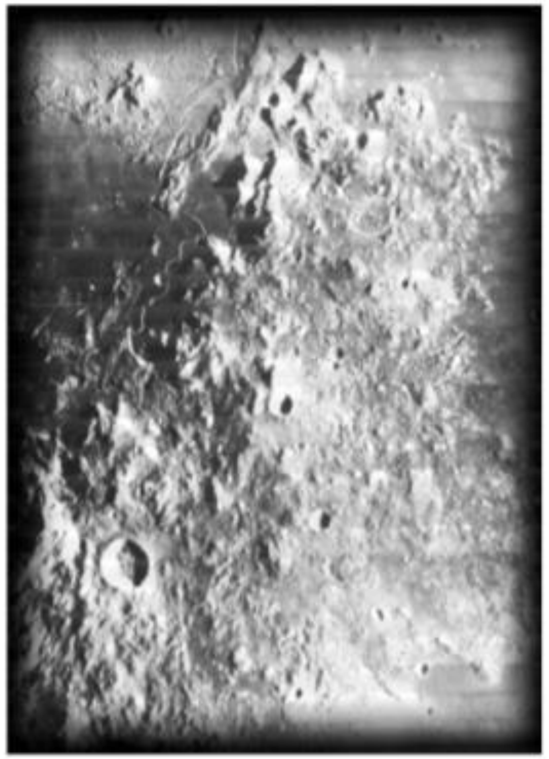




$|F(u,v)|$



remove peaks



join lines removed

contents

- Noise and filtering
- Frequency representation
- **Decimation**
- Interpolation

Imag

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

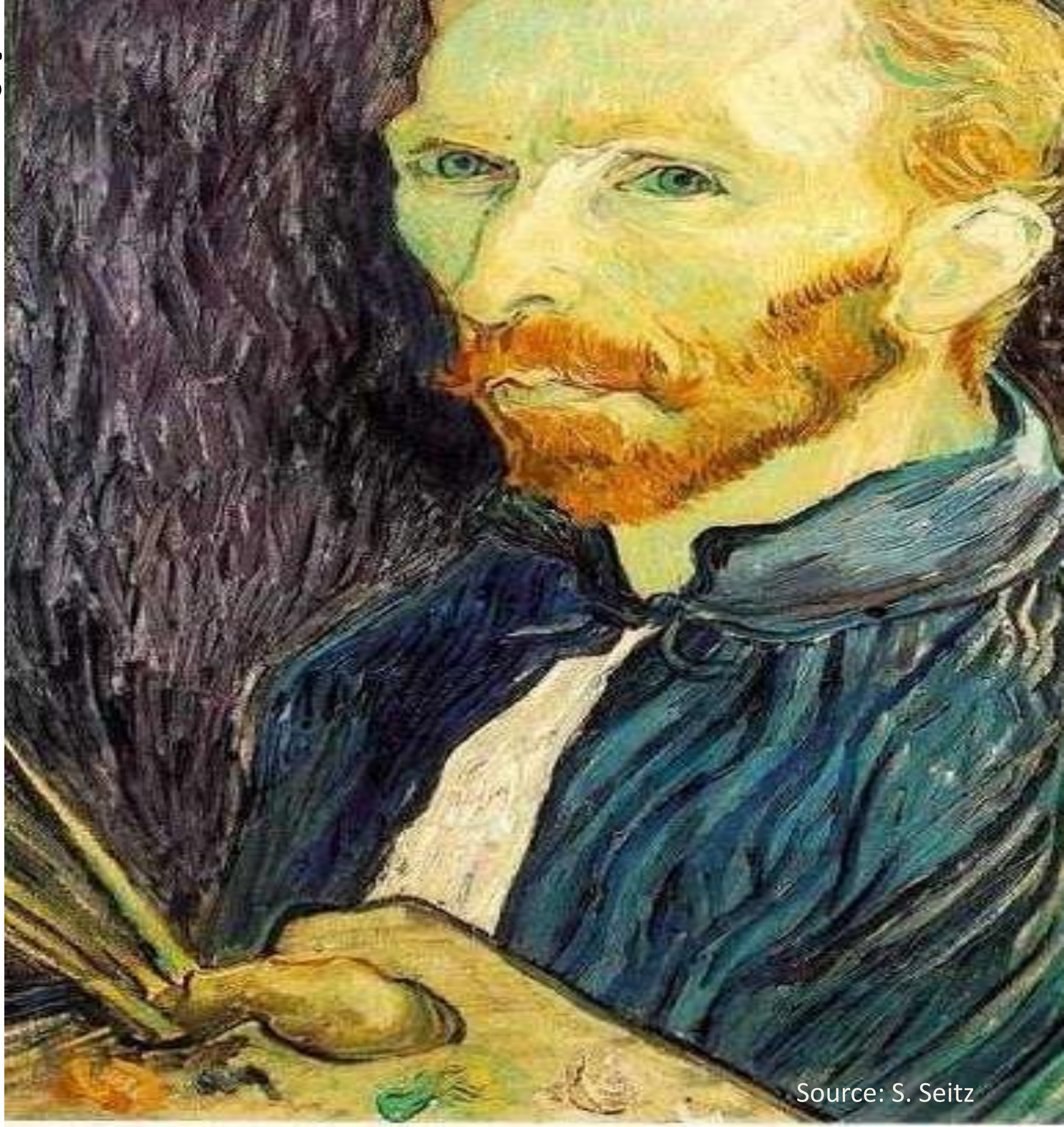
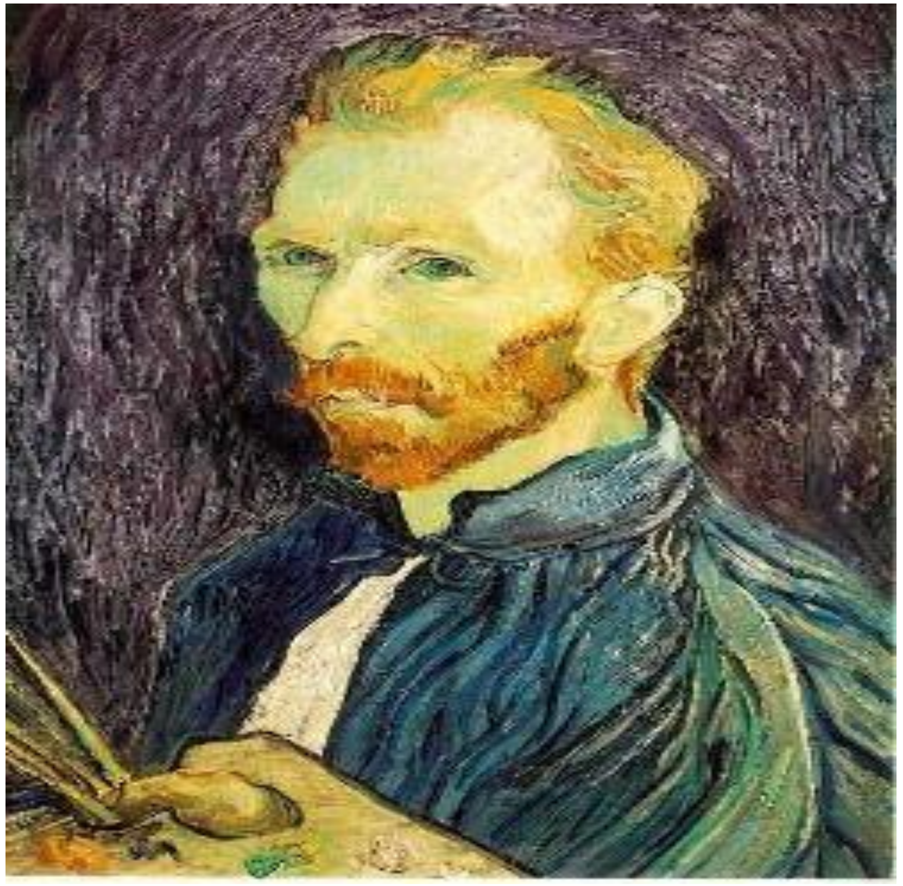
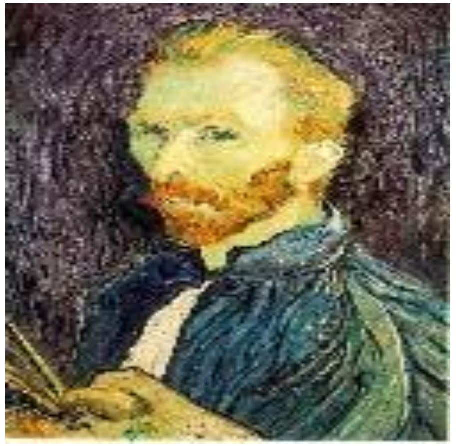


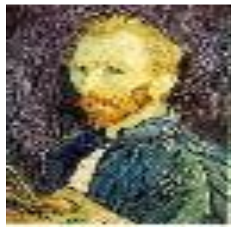
Image sub-sampling

- Throw away every other row and column to create a 1/2 size image.
  - Called **image sub-sampling** or **decimation.**
- Naïve subsampling examples:





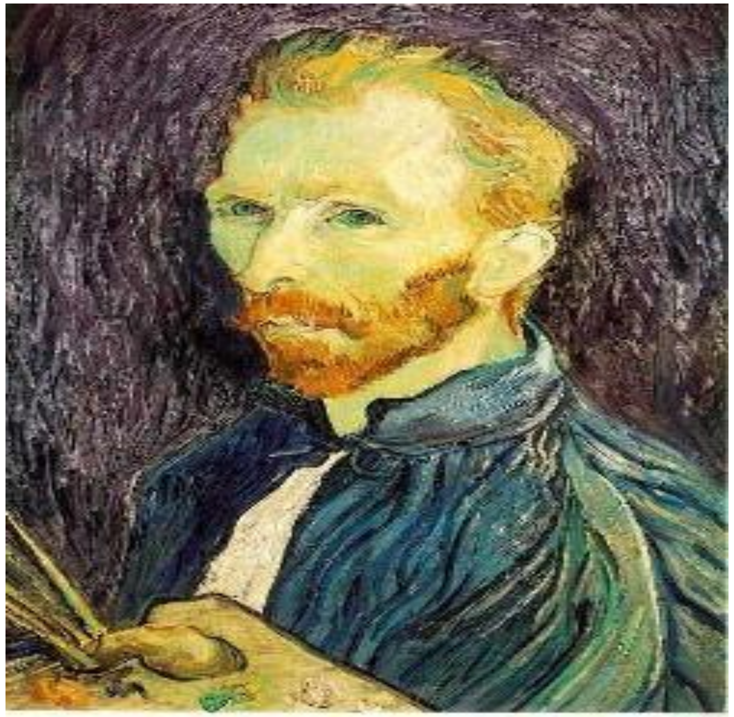
1/4

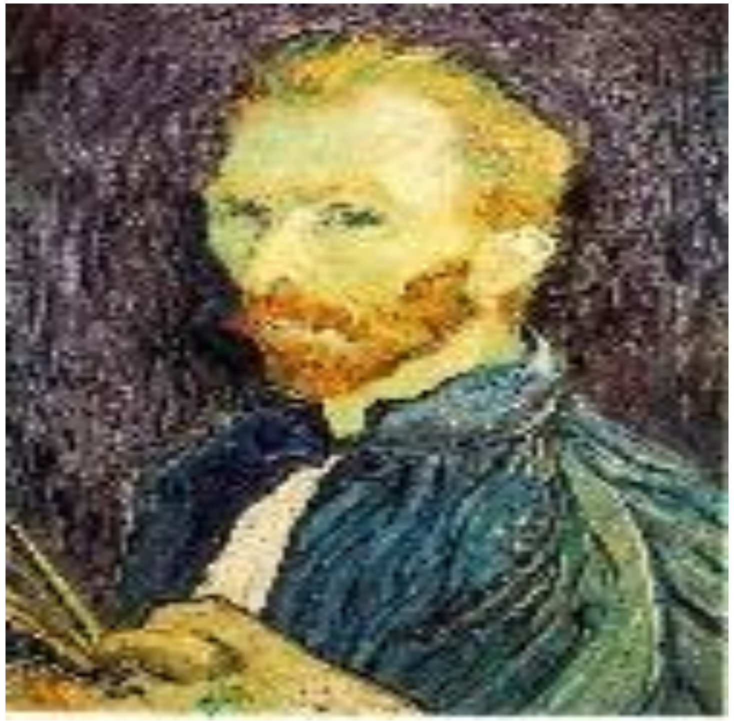


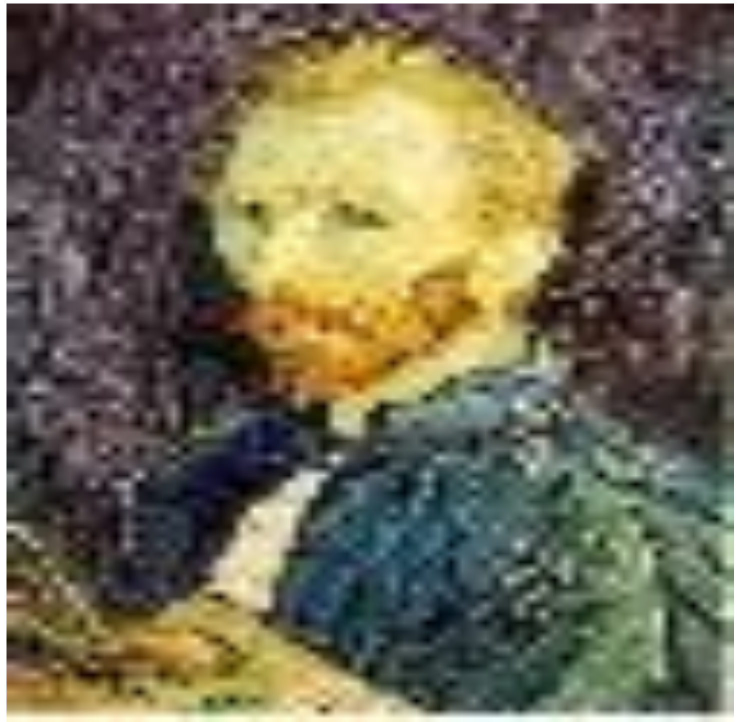
1/8

Source: S. Seitz

Image sub-sampling

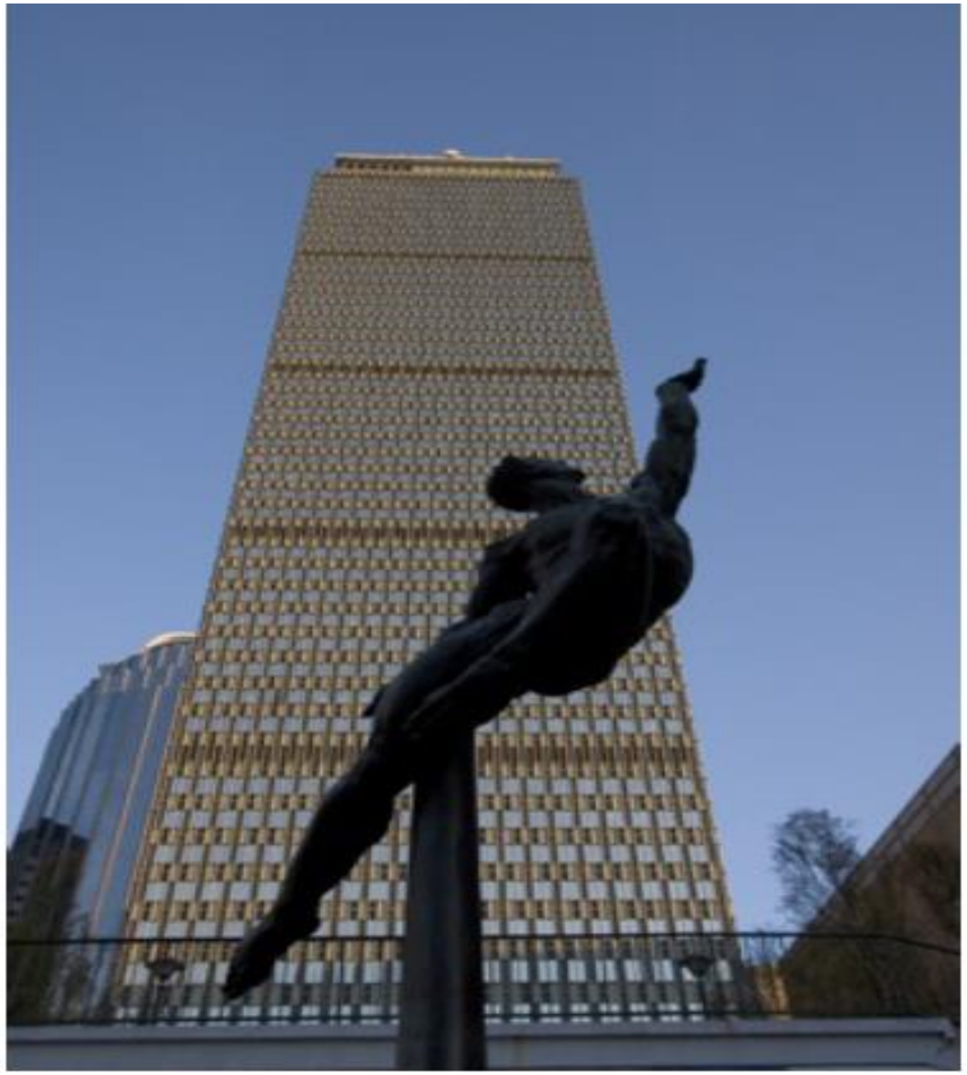

1/2

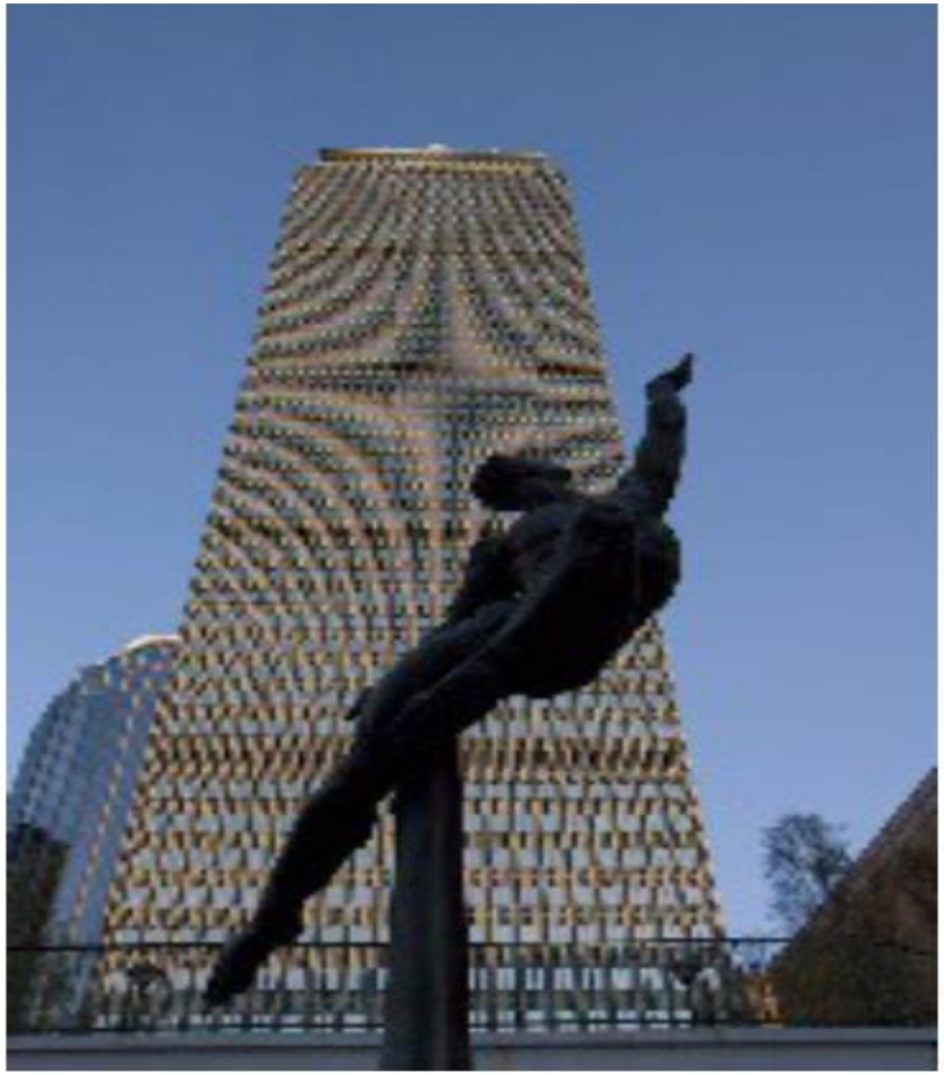

1/4 (2x zoom)


1/8 (4x zoom)

Source: S. Seitz

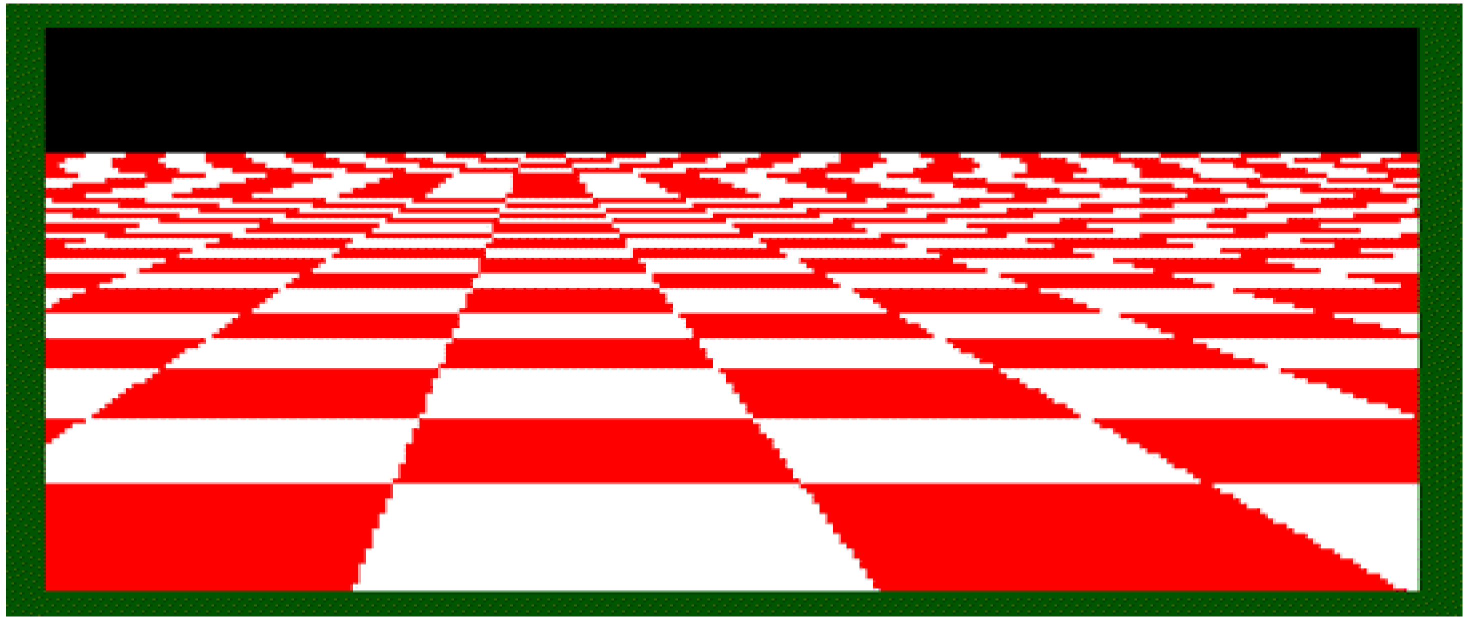
Image sub-sampling





Source: F. Durand

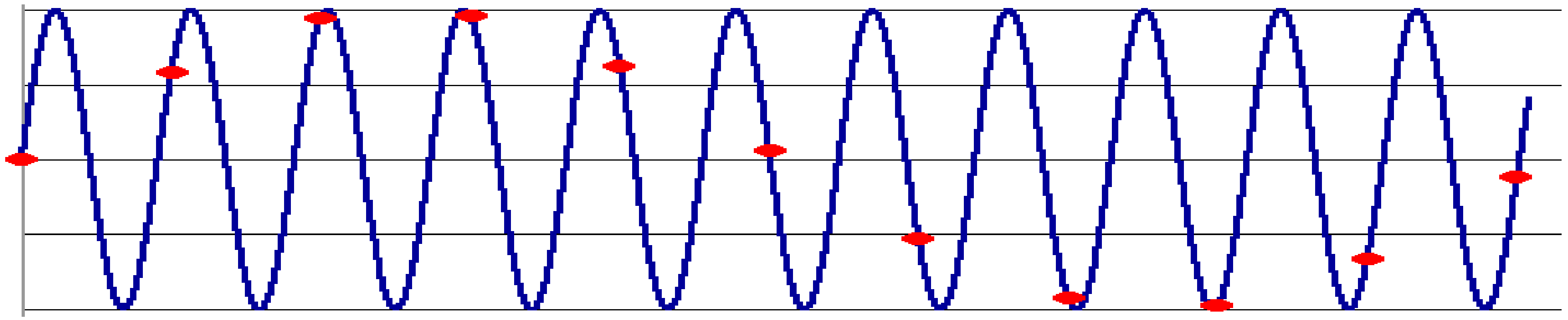
Even worse for synthetic images



Source: L. Zhang

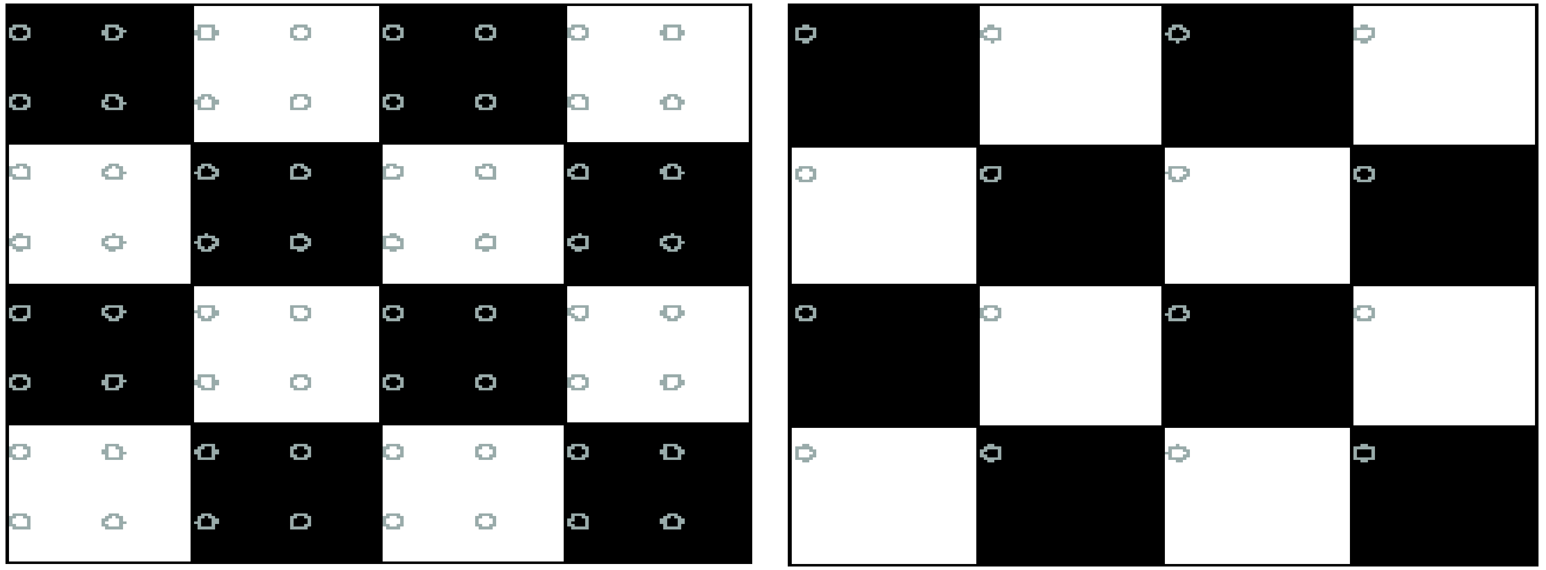
Aliasing

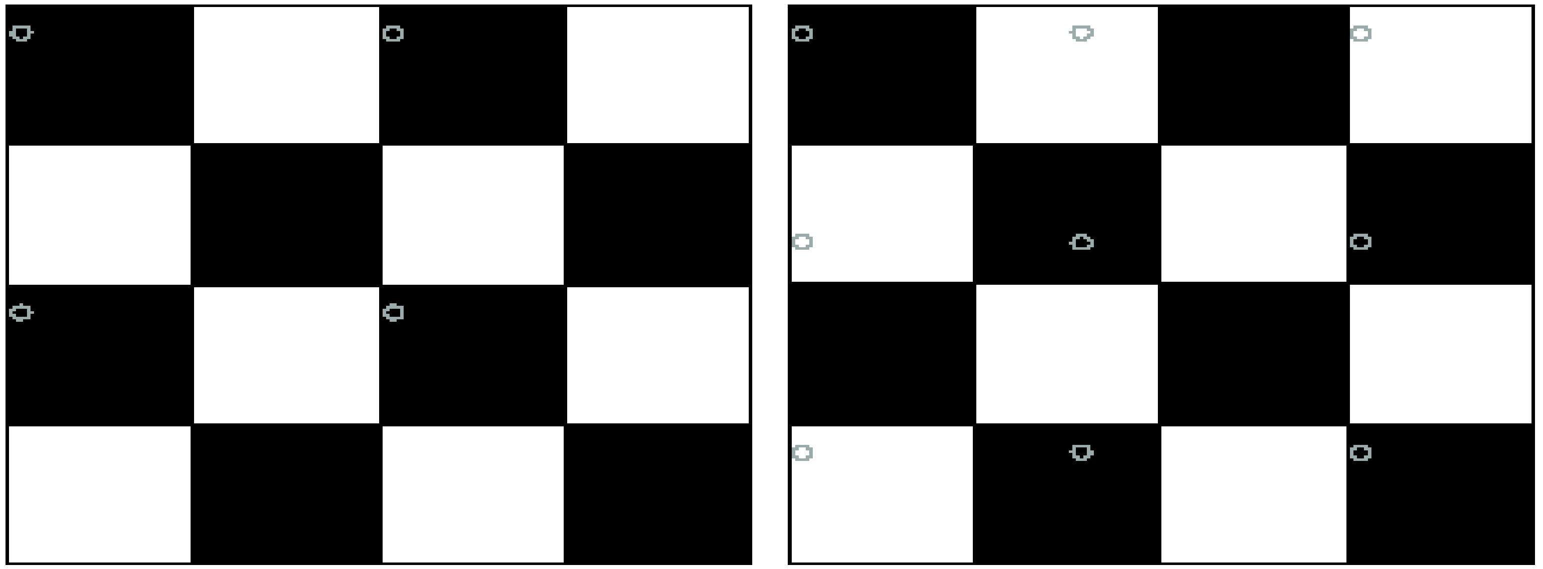
- Occurs when the sampling rate is not high enough to capture the amount of details in the image.
- Can give the wrong signal/image—an *alias.*
- To do sampling right, need to understand the structure of your signal/image
- To avoid aliasing:
  - **sampling rate ≥ 2 * max frequency** of the image.
  - This minimum sampling rate without aliasing is called the **Nyquist rate.**



Source: L. Zhang

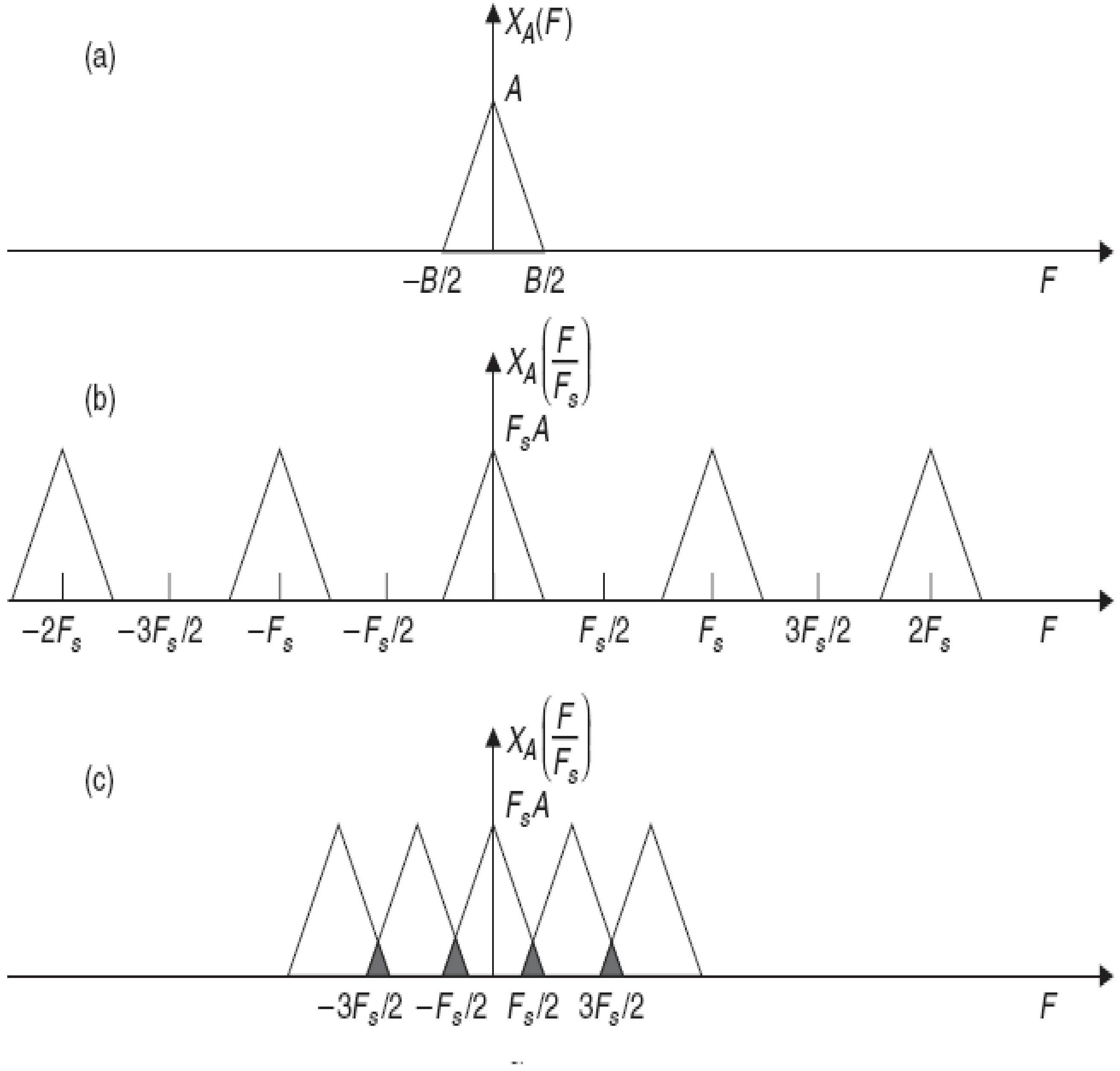
Nyquist limit – 2D example


Good sampling


Bad sampling

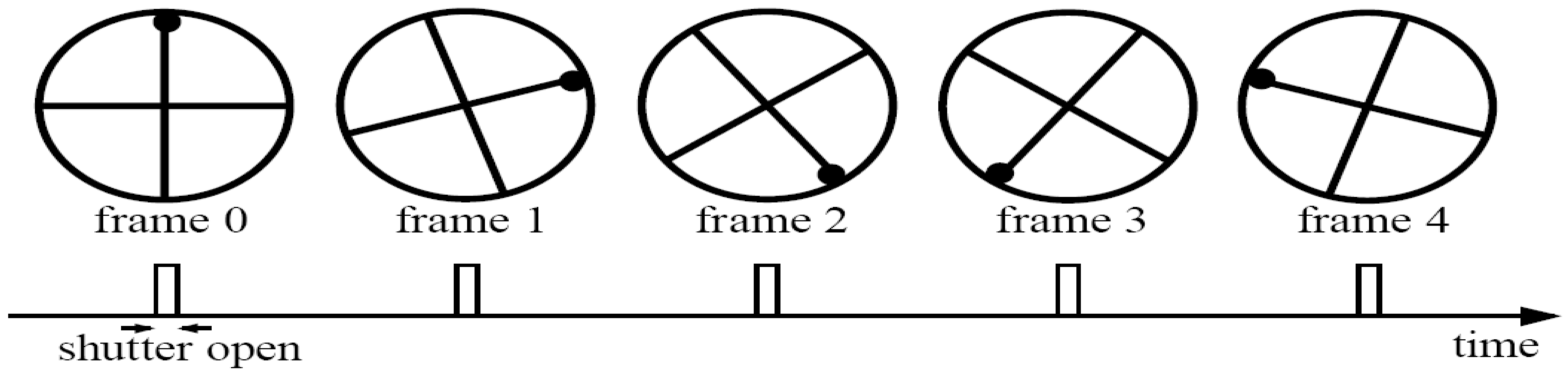
Nyquist limit- frequency response

- Original frequency representation of signal.
- Regular sampling above nyquist rate- can recreate the original frequencies of the image. (copies are from sampling).
- Sampling below nyquist- original frequencies are destroyed due to the copies overlap- **this is the aliasing**.



Example: wagon-wheel effect

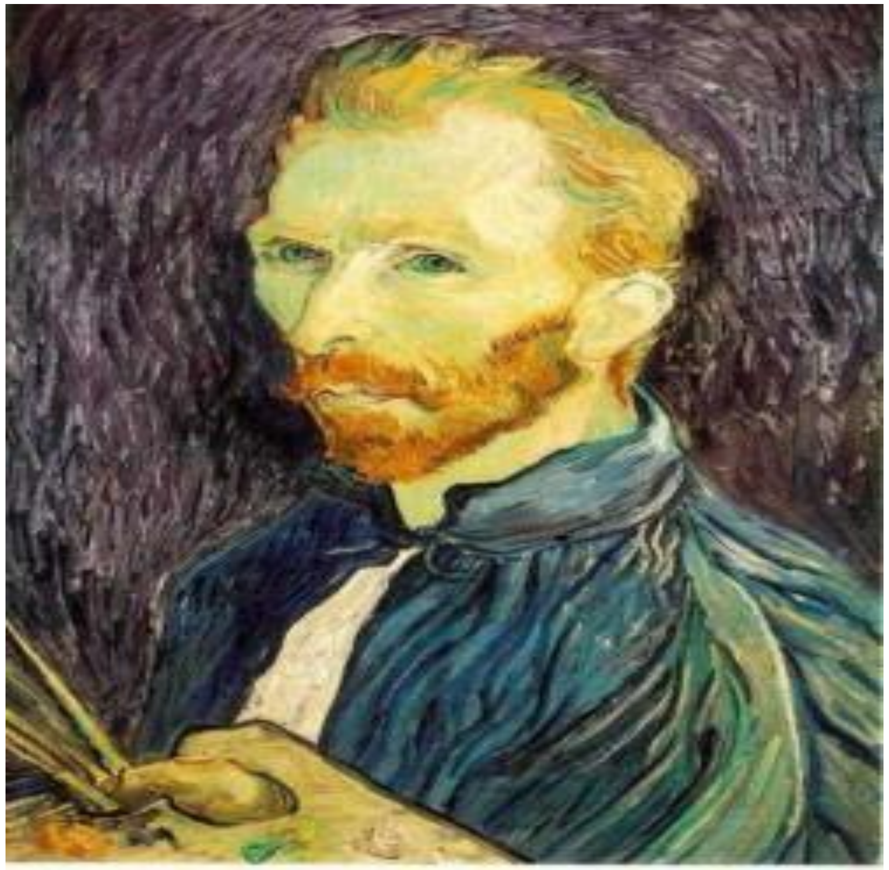
- An example of sub sampling in time domain (instead of spatially like before).



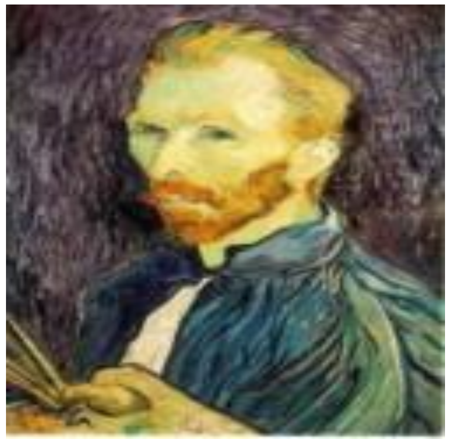
- Without the dot, the wheel appears to be rotating slowly backwards (counterclockwise).
- https://en.wikipedia.org/wiki/File:Propeller_strobe.ogv
- https://en.wikipedia.org/wiki/File:The_wagon-wheel_effect.ogv

Gaussian pre-filtering

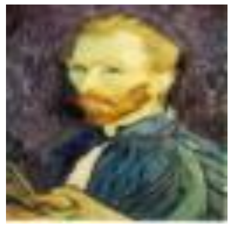
- Solution: filter the image, *then* subsample



Gaussian 1/2



G 1/4

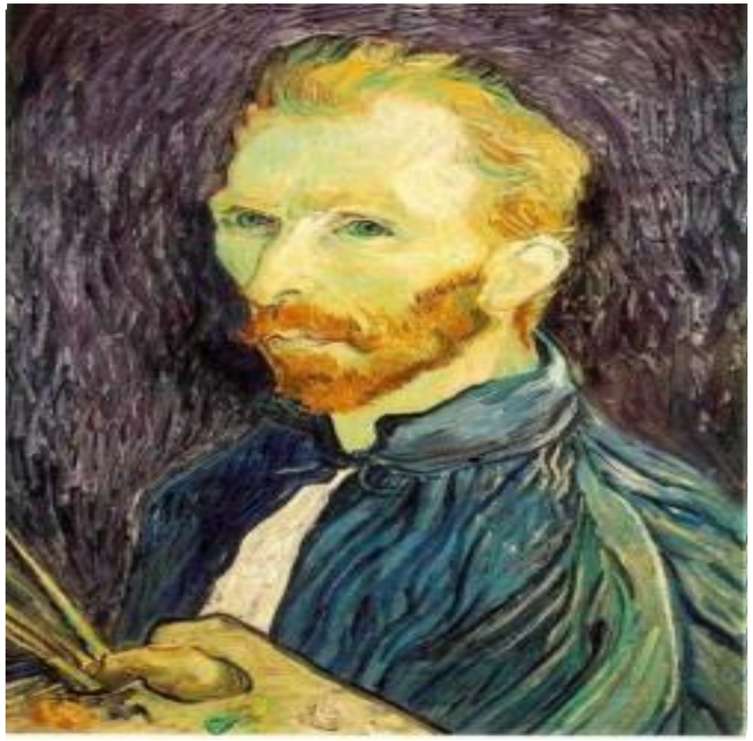


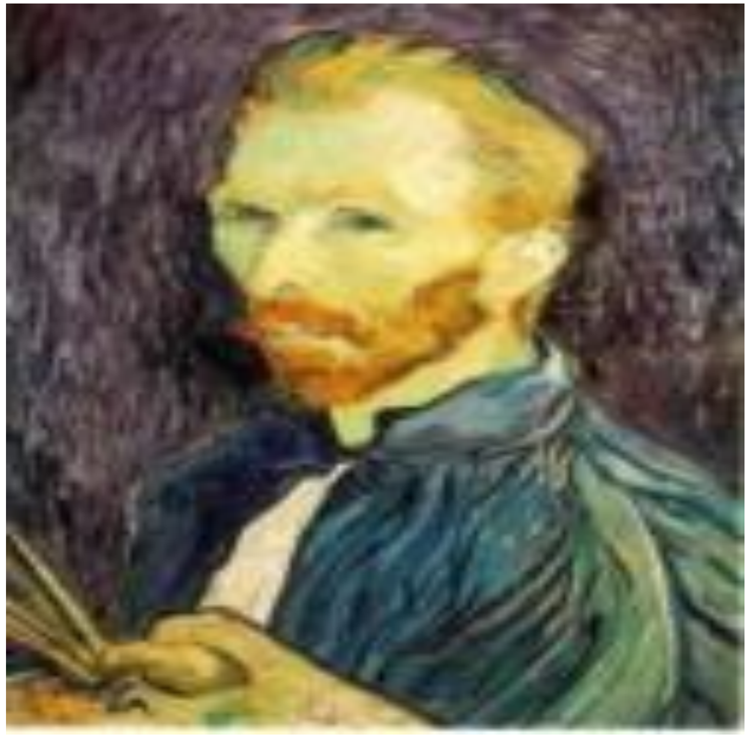
G 1/8

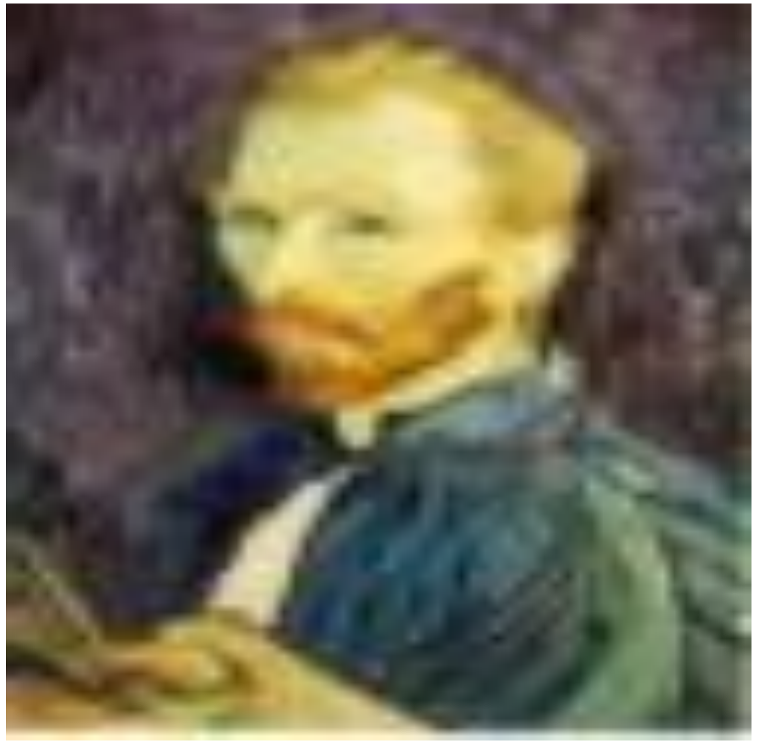
Source: S. Seitz

Subsampling with Gaussian pre-filtering

- Solution: filter the image, *then* subsample

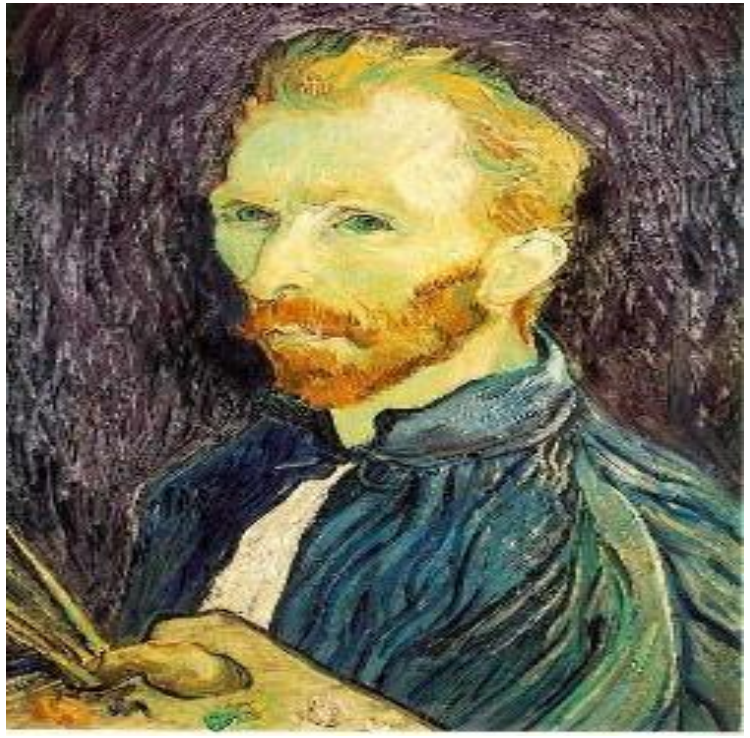

Gaussian 1/2

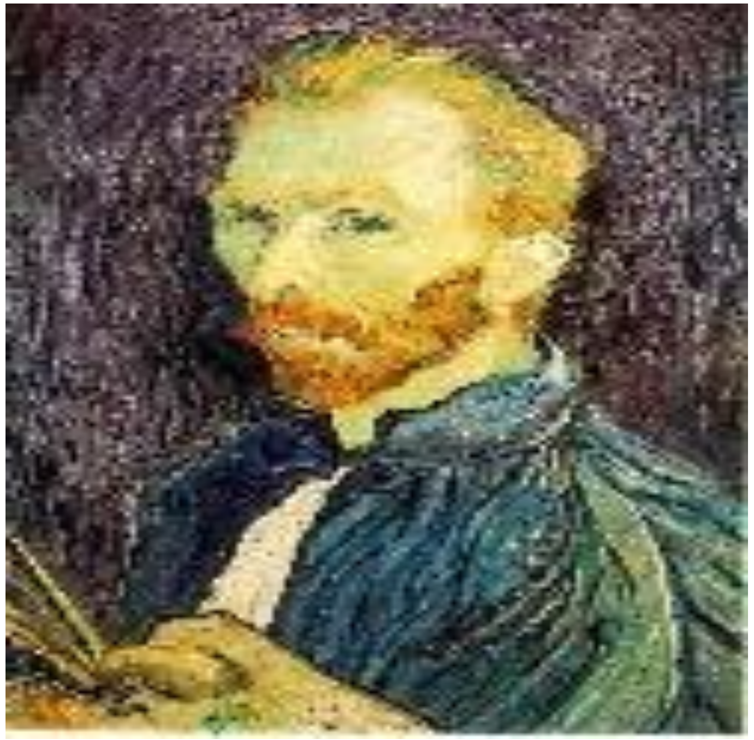

G 1/4

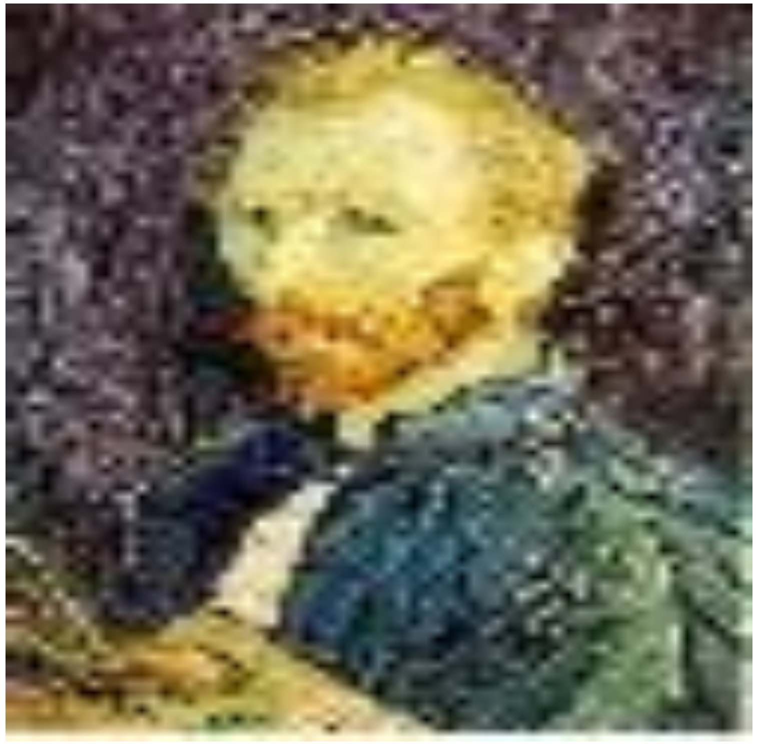

G 1/8

Source: S. Seitz

Compare with...

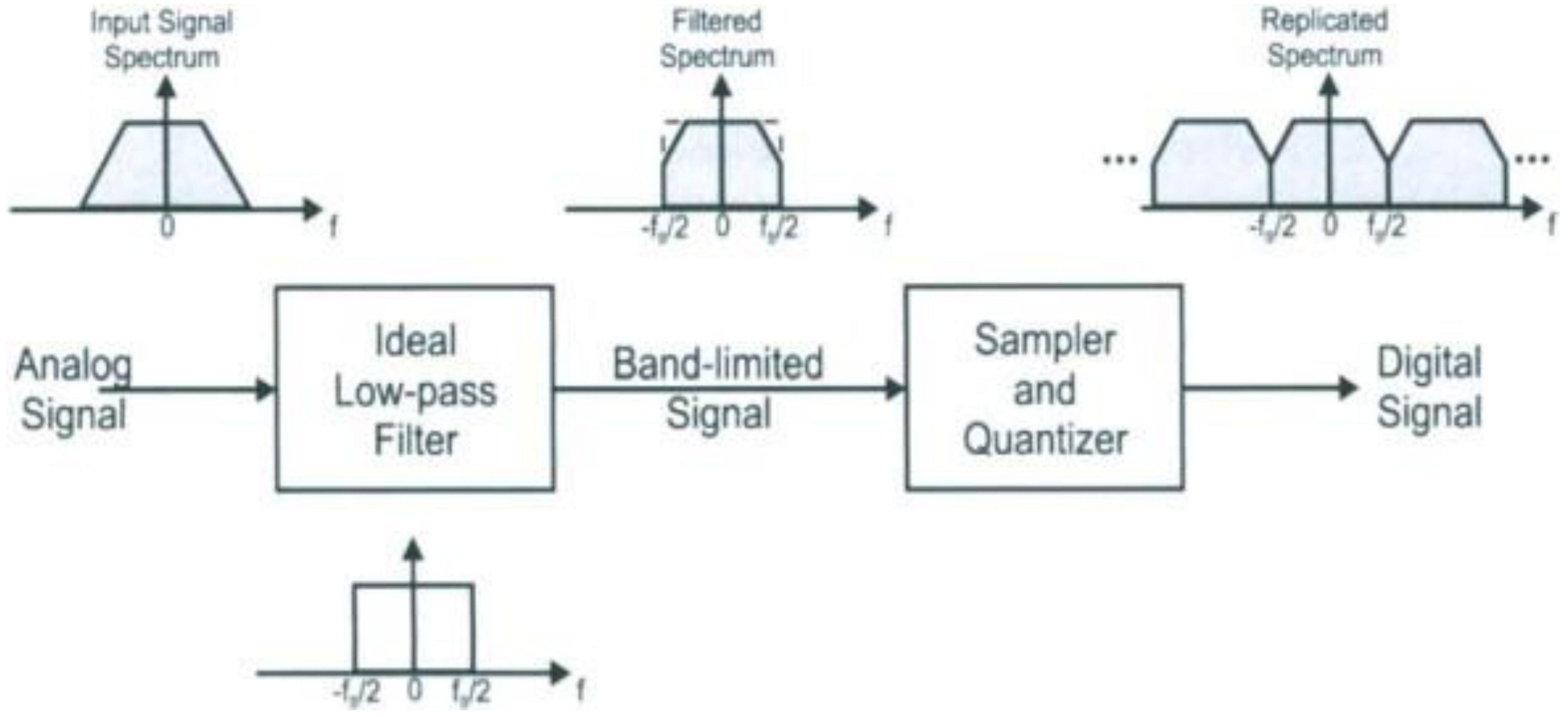

1/2


1/4 (2x zoom)


1/8 (4x zoom)

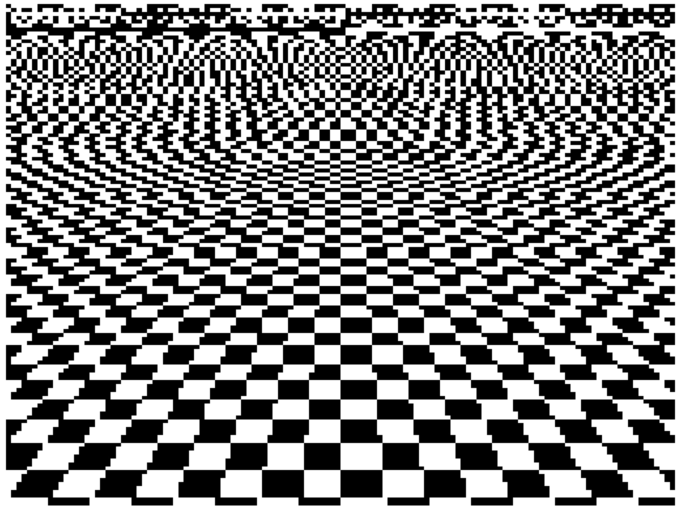
Source: S. Seitz

Low pass filtering- frequency response

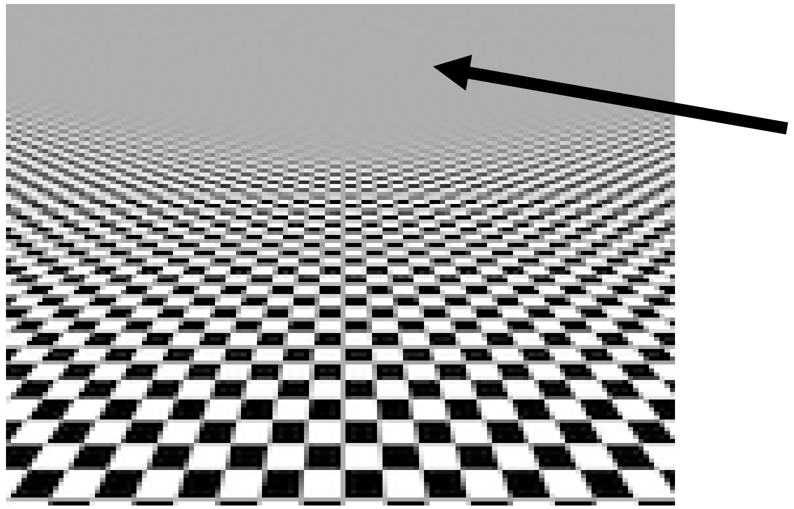


Back to the checkerboard

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



Naïve subsampling

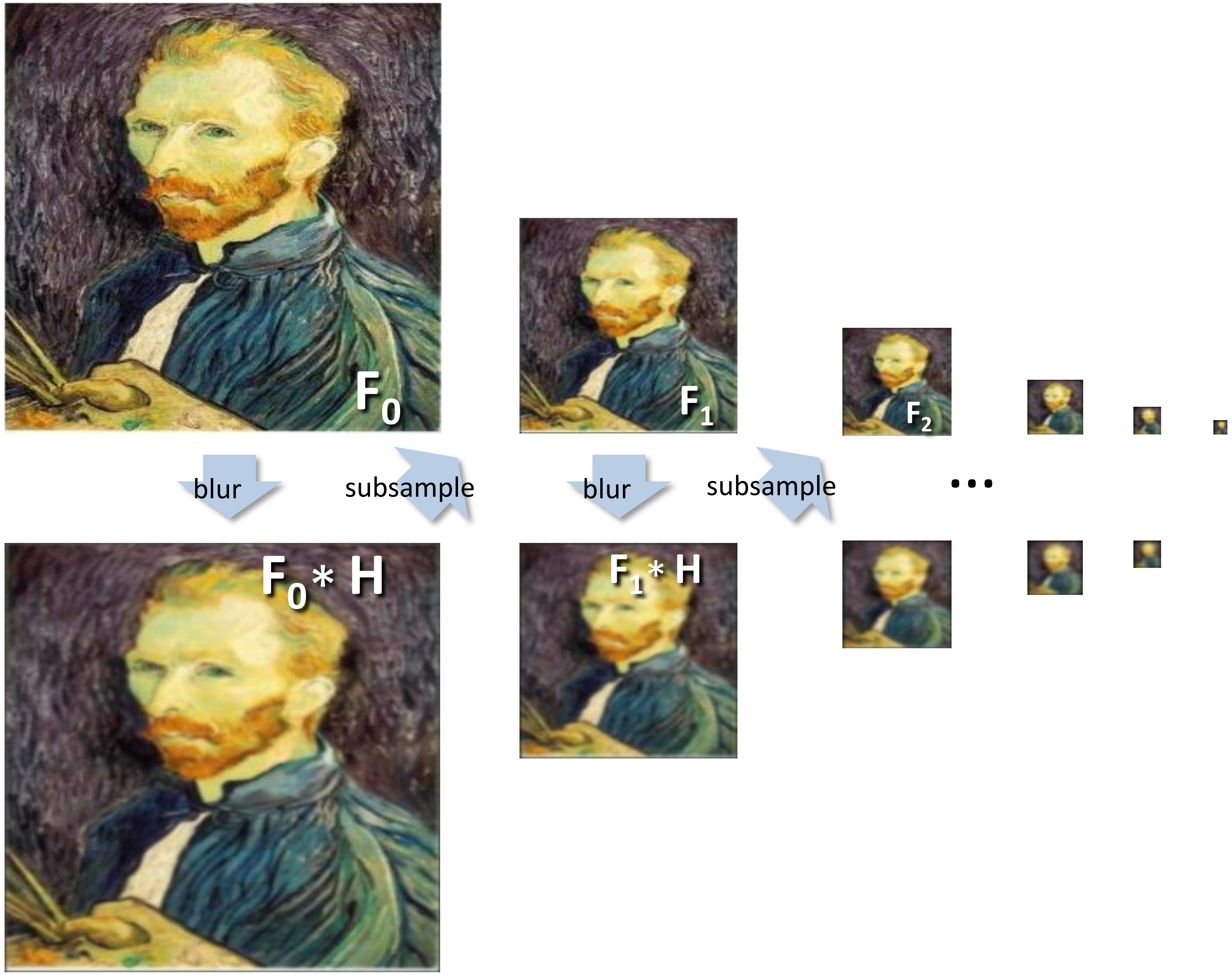


Proper prefiltering
(“antialiasing”)

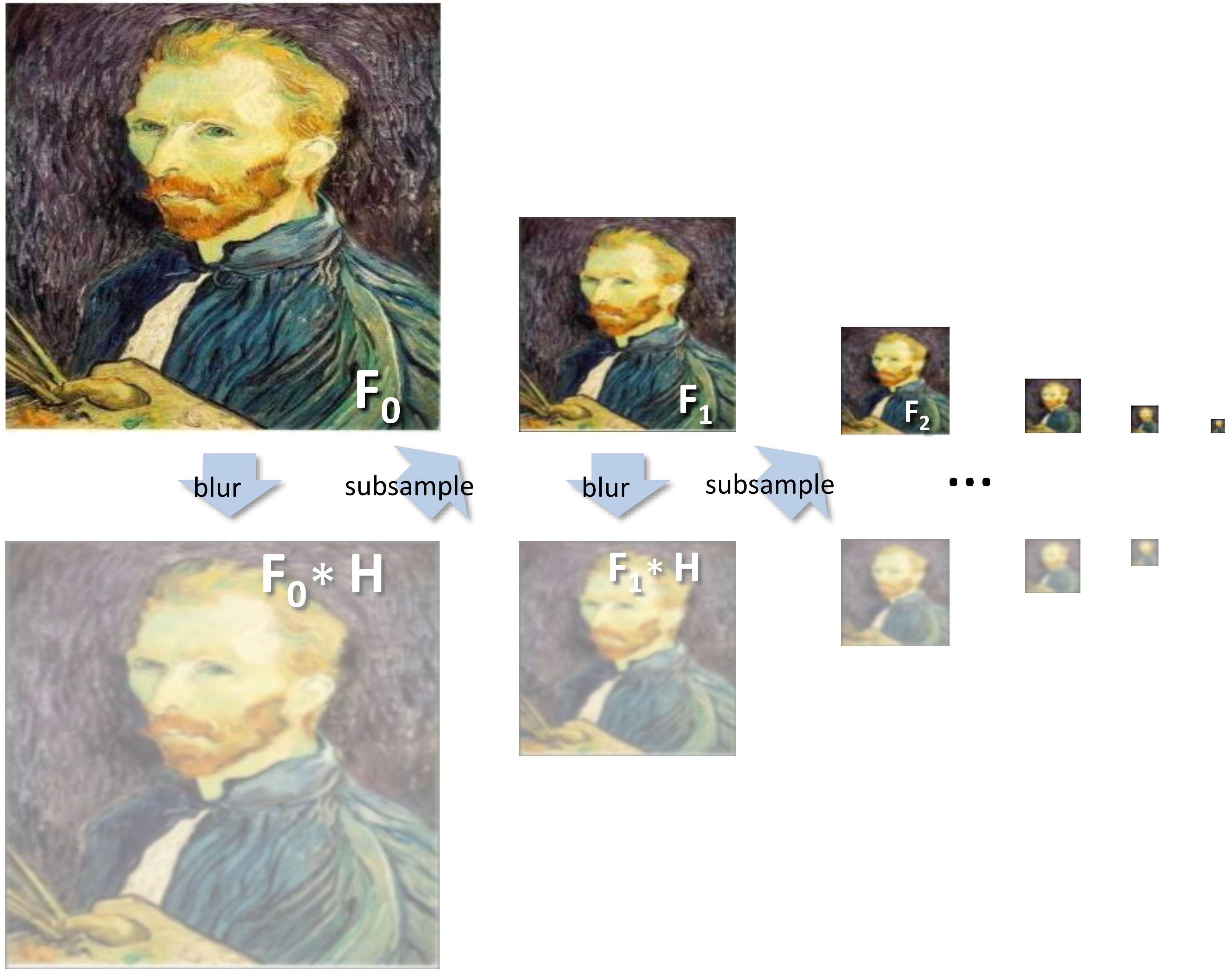
Image turns grey! (Average of black and white squares, because each pixel contains both.)

Gaussian pre-filtering

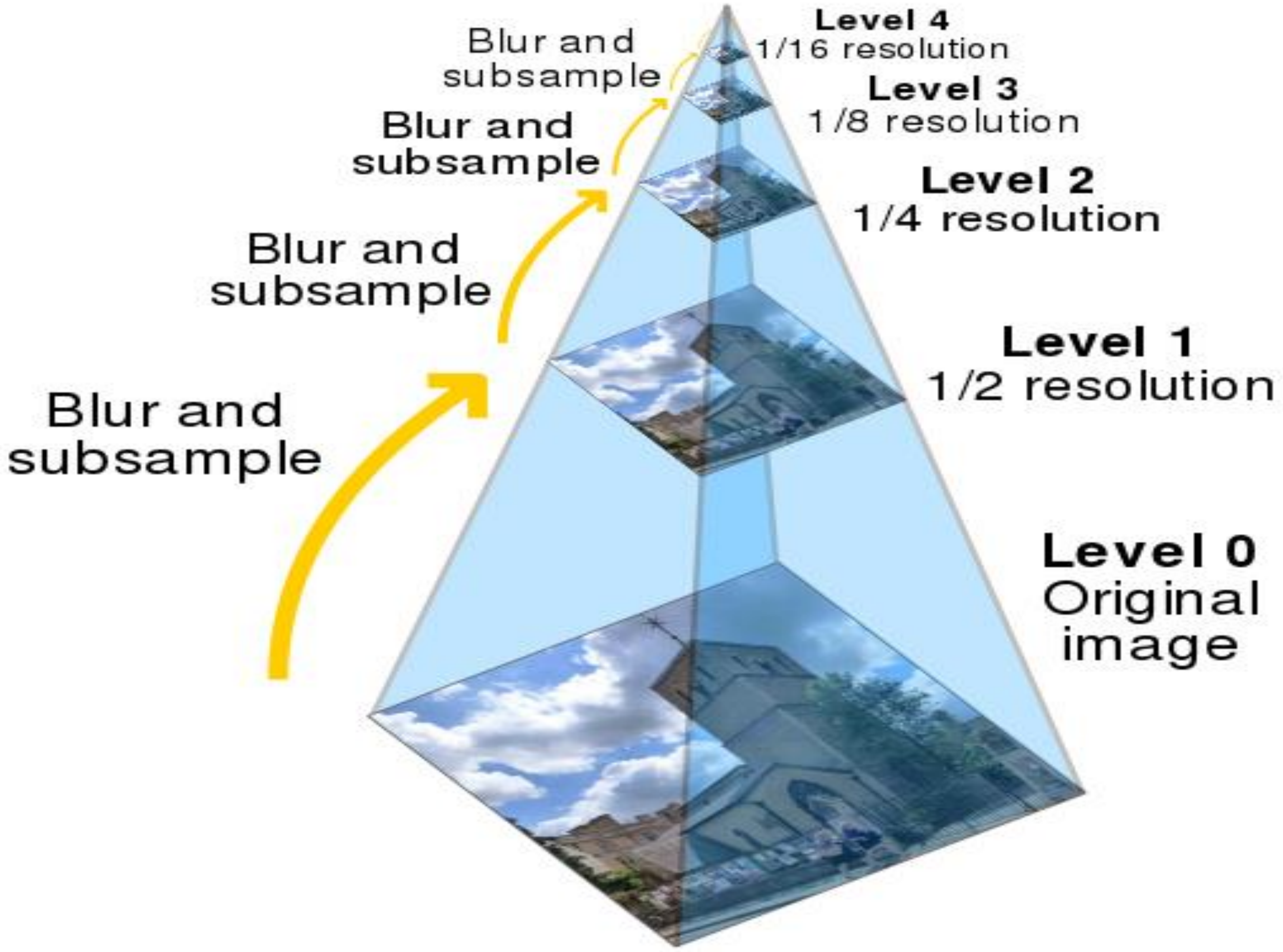
- Solution: filter the image, *then* subsample



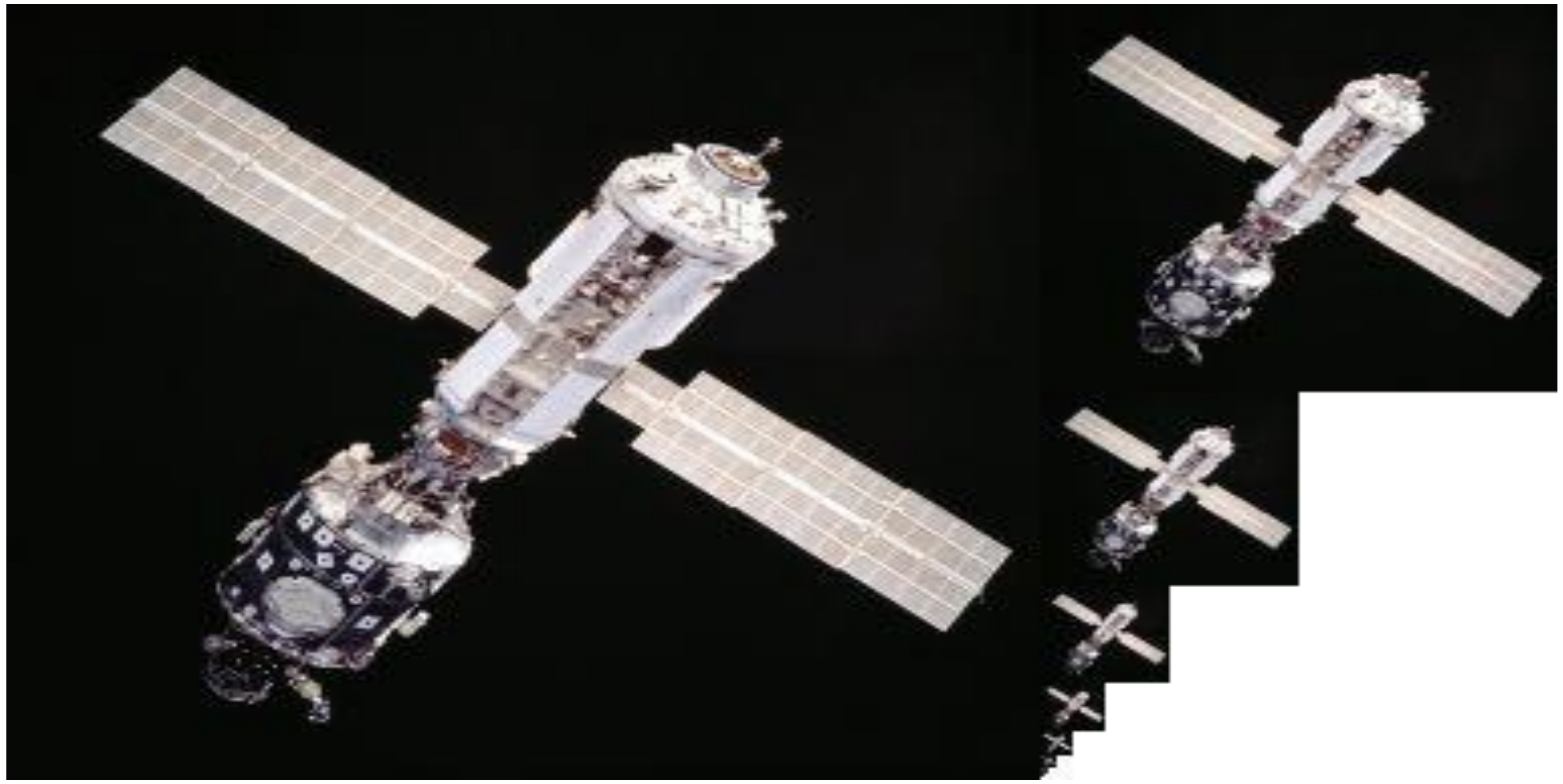
Gaussian pyramid



Gaussian pyramid



Gaussian pyramid

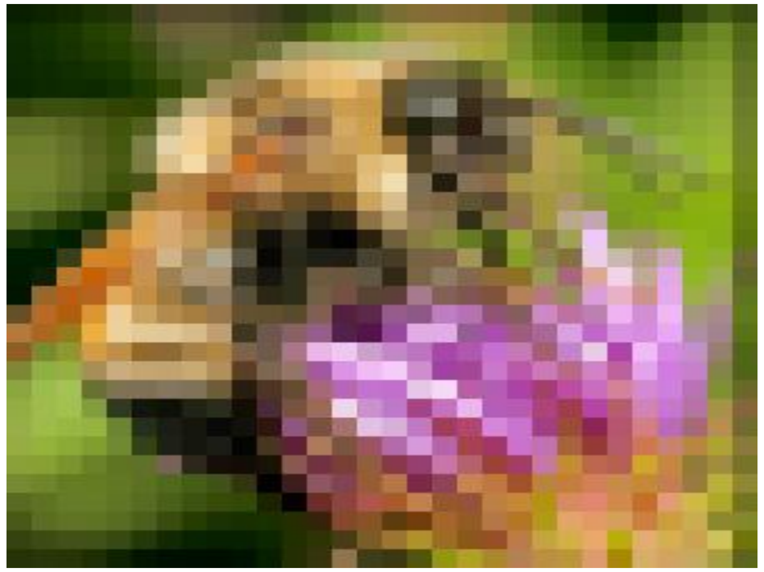


contents

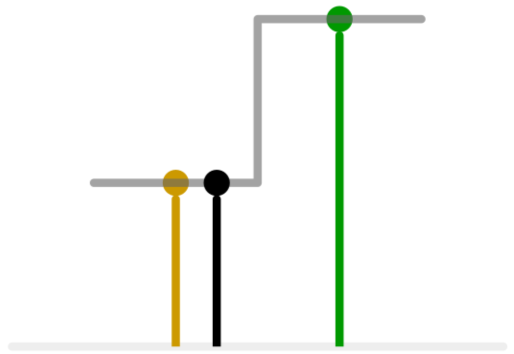
- Noise and filtering
- Frequency representation
- Decimation
- **Interpolation**

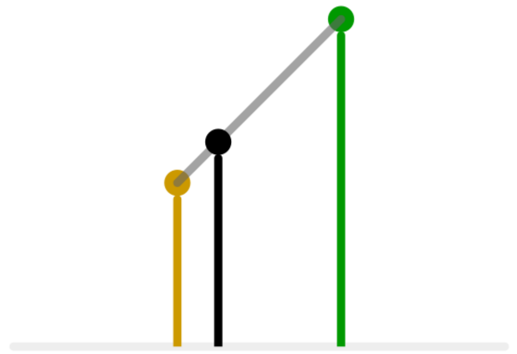
Upsampling

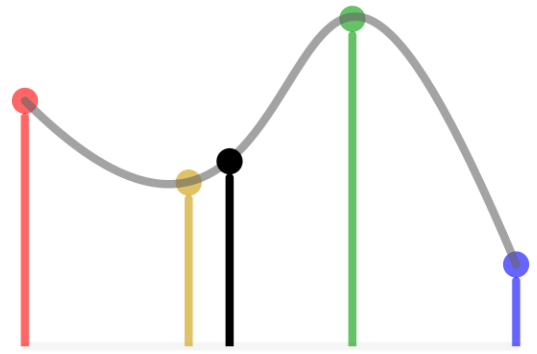
- This image is too small for this screen:
- How can we make it 10 times as big?
- Simplest approach: repeat each row and column 10 times ("**Nearest neighbor interpolation**")
- This operation is known as **upsampling** or **interpolation.**

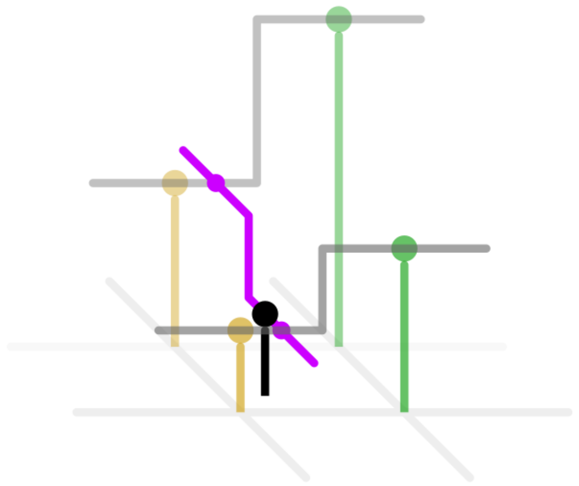


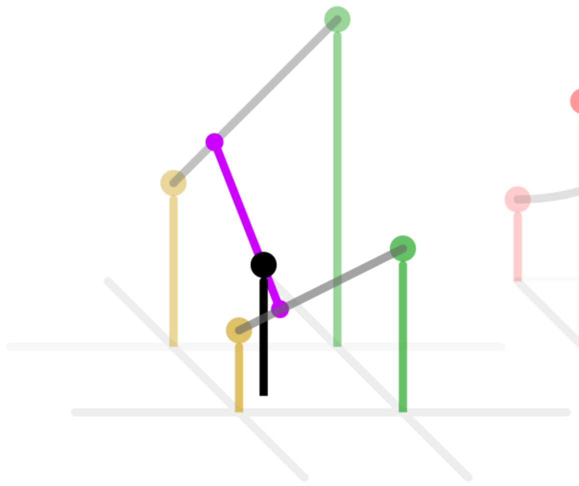
Upsampling

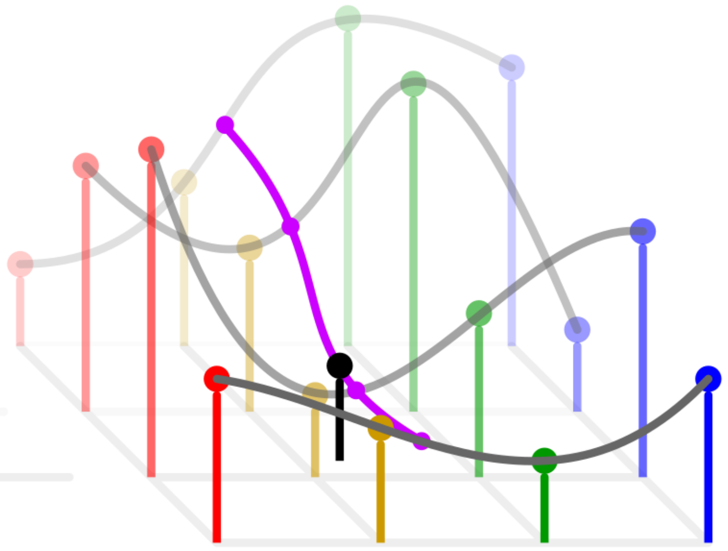

1D nearest-neighbour


Linear

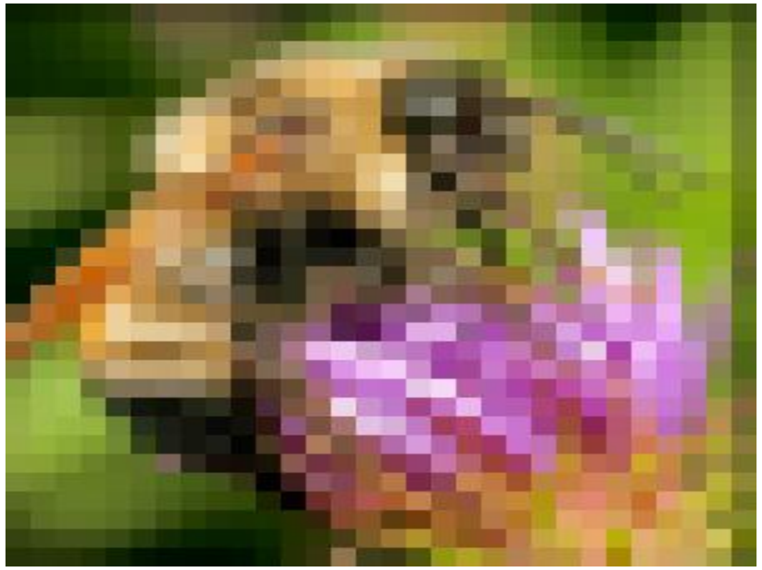

Cubic

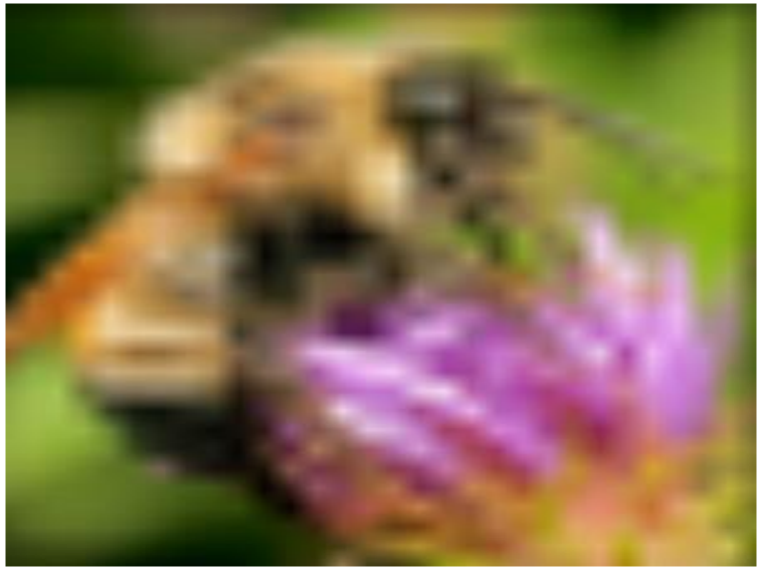

2D nearest-neighbour

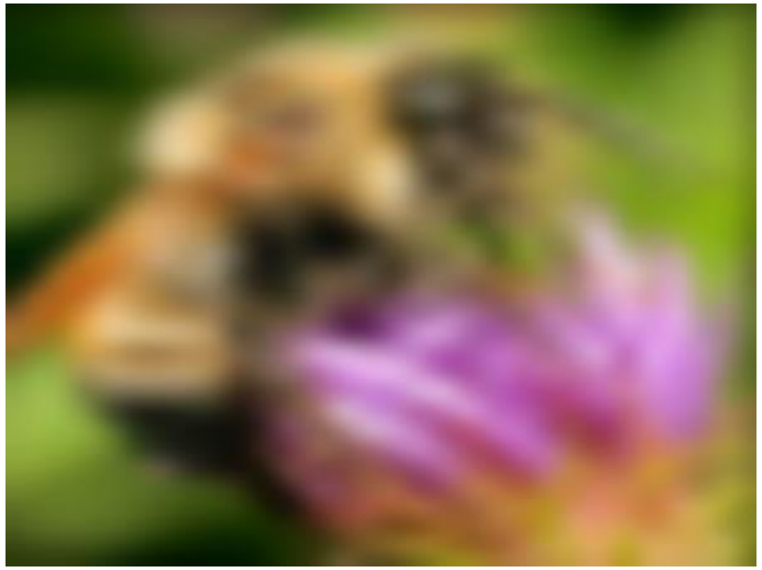

Bilinear


Bicubic

Upsampling


Nearest-neighbor interpolation


Bilinear interpolation


Bicubic interpolation

Upsampling

- Also used for image warping