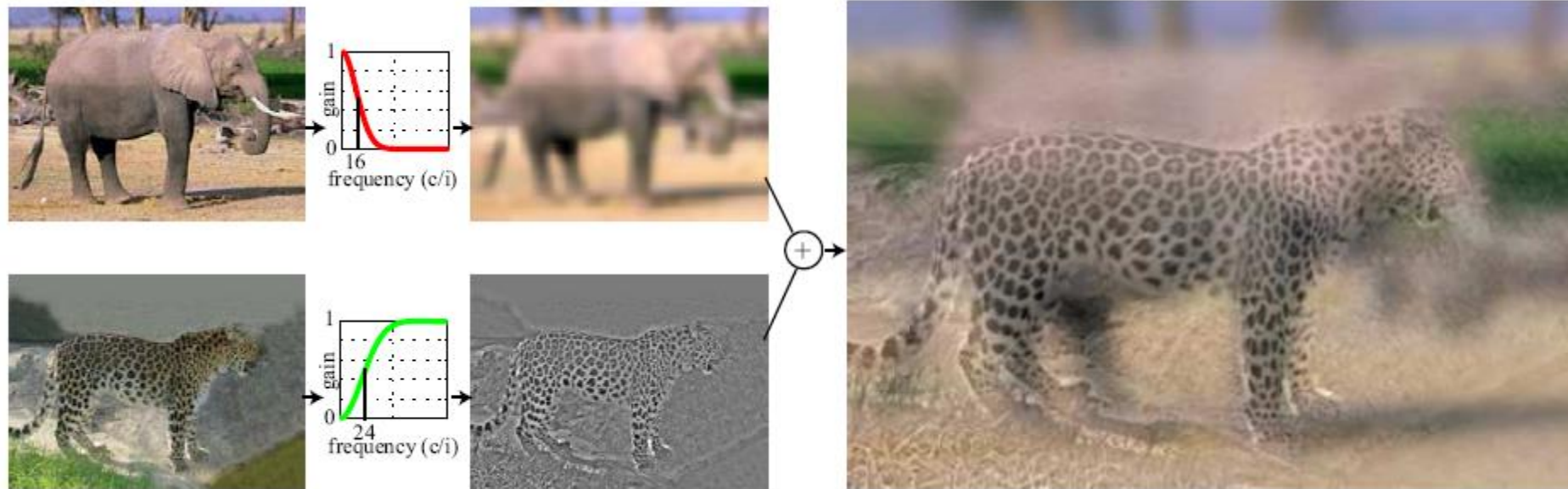


The blue and green colors are actually the same



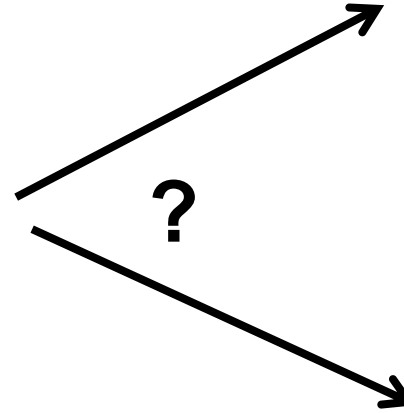
<http://blogs.discovermagazine.com/badastronomy/2009/06/24/the-blue-and-the-green/>

Hybrid Images



- A. Oliva, A. Torralba, P.G. Schyns, [“Hybrid Images,”](#) SIGGRAPH 2006

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





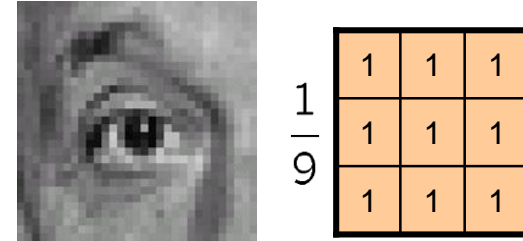


Thinking in Frequency



Recap of Filtering

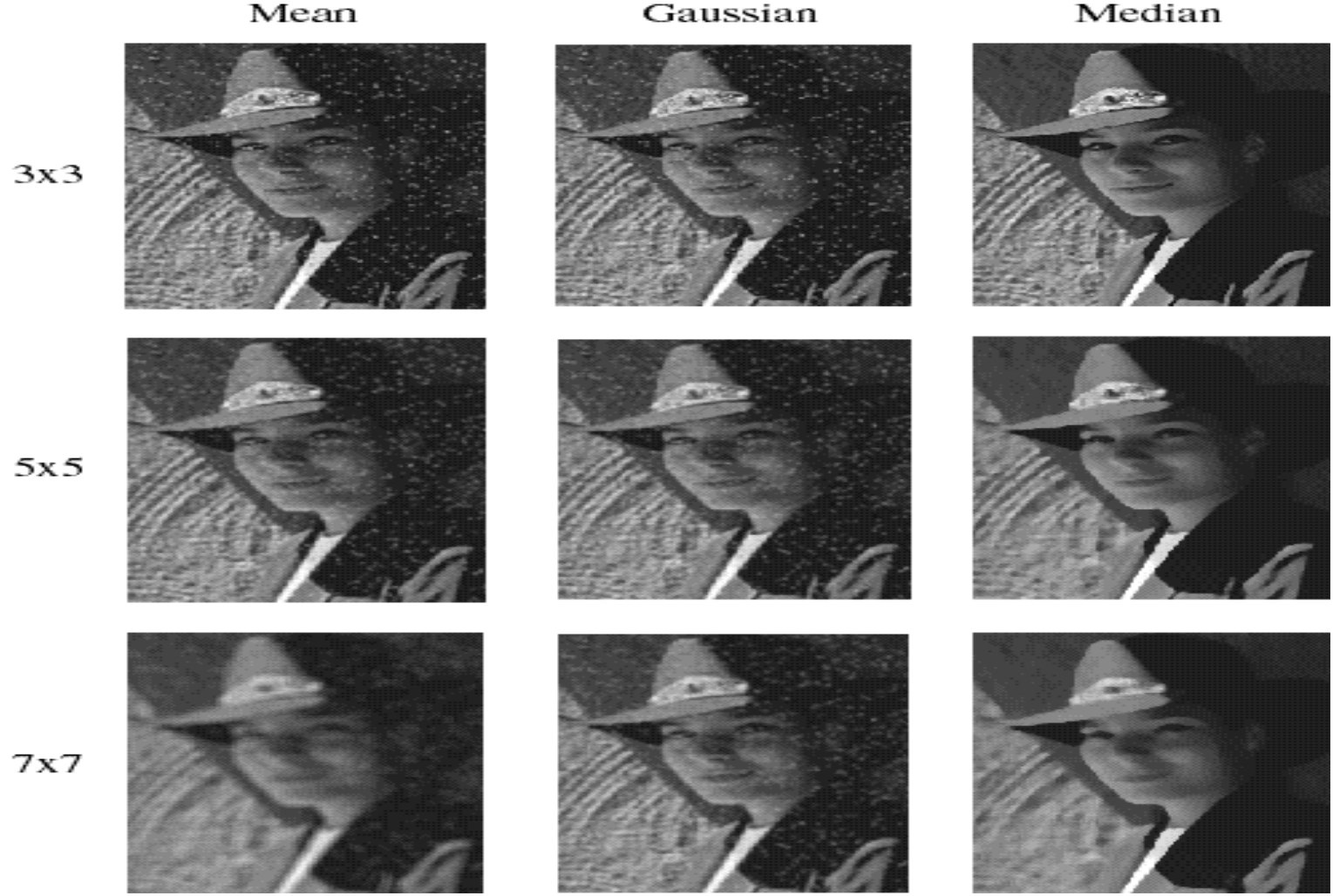
- Linear filtering is dot product at each position
 - Not a matrix multiplication
 - Can smooth, sharpen, translate (among many other uses)
- Be aware of details for filter size, extrapolation, cropping



Median filters

- A **Median Filter** operates over a window by selecting the median intensity in the window.
- What advantage does a median filter have over a mean filter?
- Is a median filter a kind of convolution?

Comparison: salt and pepper noise



Review: questions

1. Write down a 3x3 filter that returns a positive value if the average value of the 4-adjacent neighbors is less than the center and a negative value otherwise
2. Write down a filter that will compute the gradient in the x-direction:

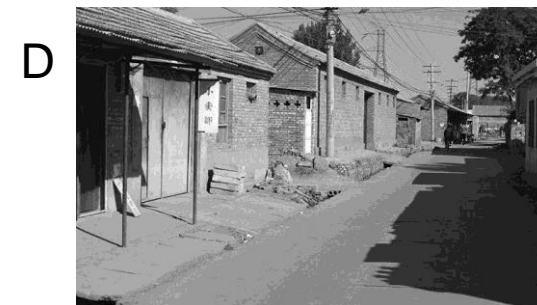
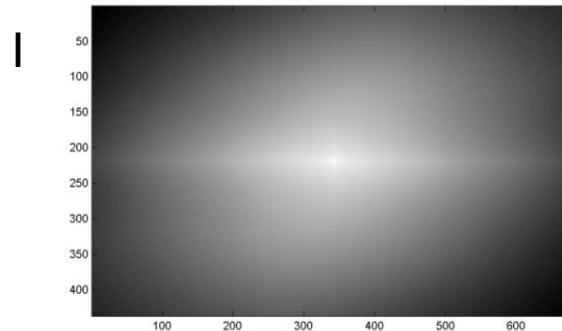
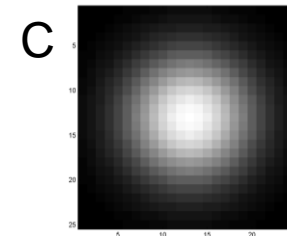
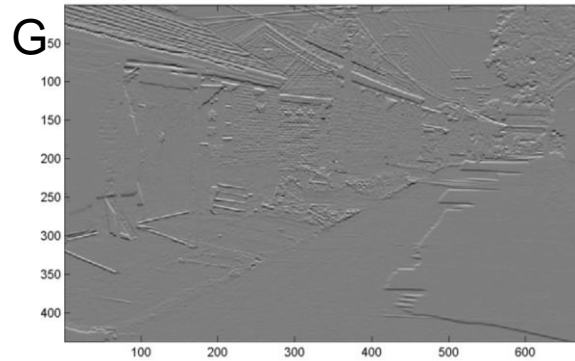
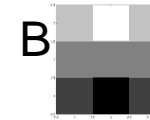
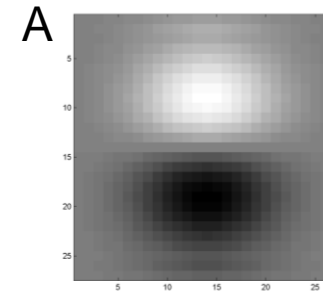
$$\text{grad}_x(y, x) = \text{im}(y, x+1) - \text{im}(y, x) \text{ for each } x, y$$

Review: questions

3. Fill in the blanks:

- a) $_ = D * B$
 b) $A = _ * _$
 c) $F = D * _$
 d) $_ = D * D$

Filtering Operator

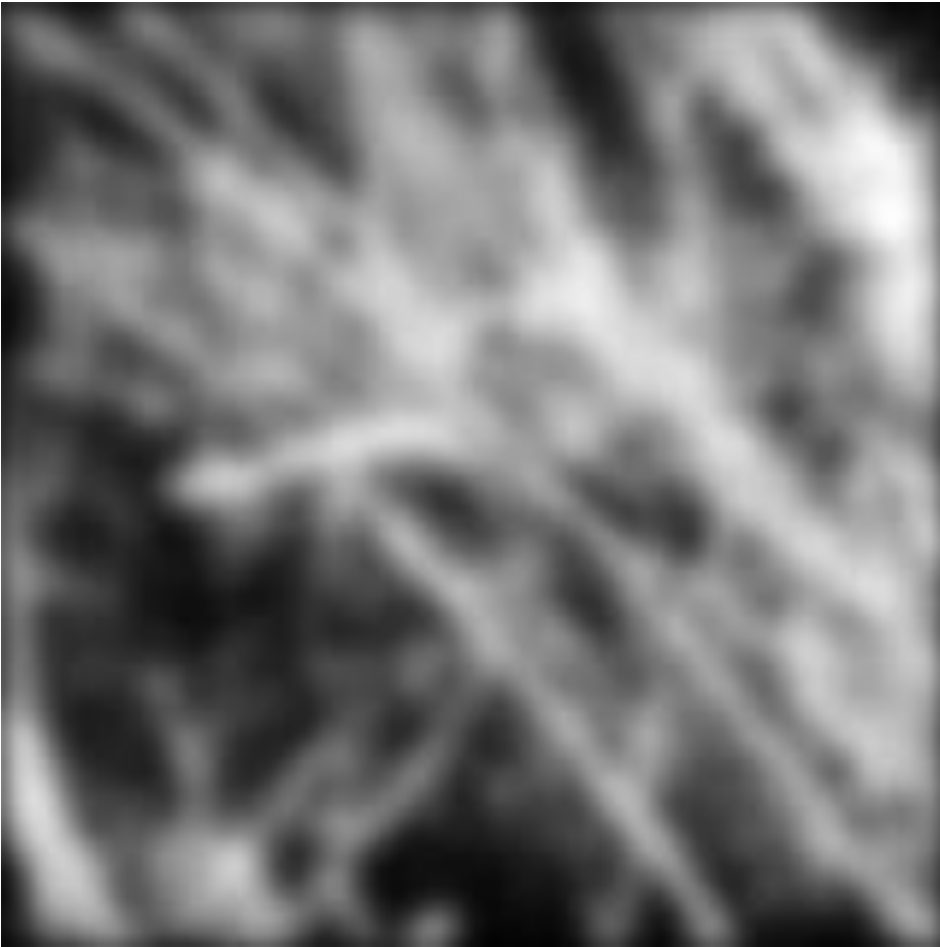


This lecture

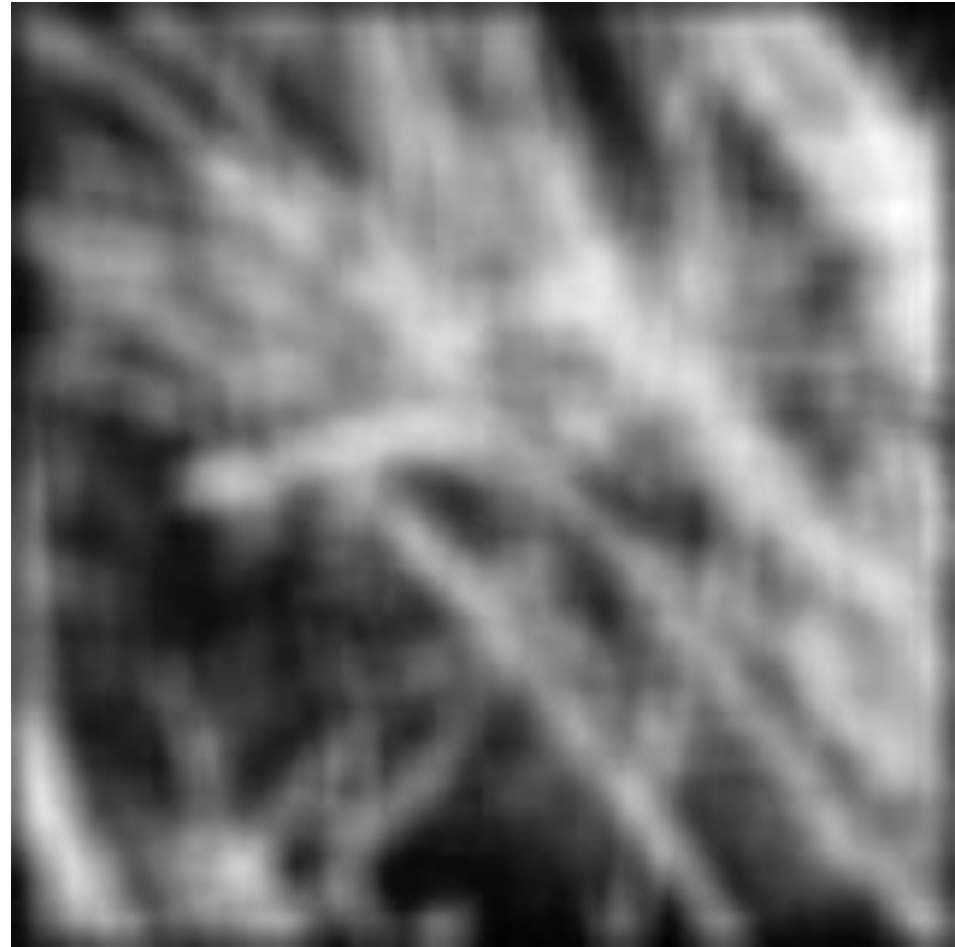
- Fourier transform and frequency domain
 - Frequency view of filtering
- Reminder: Read your textbook
 - Today's lecture covers material in 3.4

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

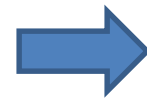
Gaussian



Box filter



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



Thinking in terms of frequency

Background: Change of Basis


← → ↻ 🏠 <https://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/video-lectures/>


EXAMS


STUDY MATERIALS

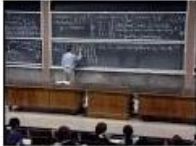
RELATED RESOURCES


DOWNLOAD COURSE MATERIALS


 » [Lecture 3: Multiplication and inverse matrices](#)


 » [Lecture 4: Factorization into \$A = LU\$](#)

 » [Lecture 5: Transposes, permutations, spaces \$\mathbb{R}^n\$](#)

 » [Lecture 6: Column space and nullspace](#)

 » [Lecture 7: Solving \$Ax = 0\$: pivot variables, special solutions](#)

 » [Lecture 8: Solving \$Ax = b\$: row reduced form \$R\$](#)

 » [Lecture 9: Independence, basis, and dimension](#)

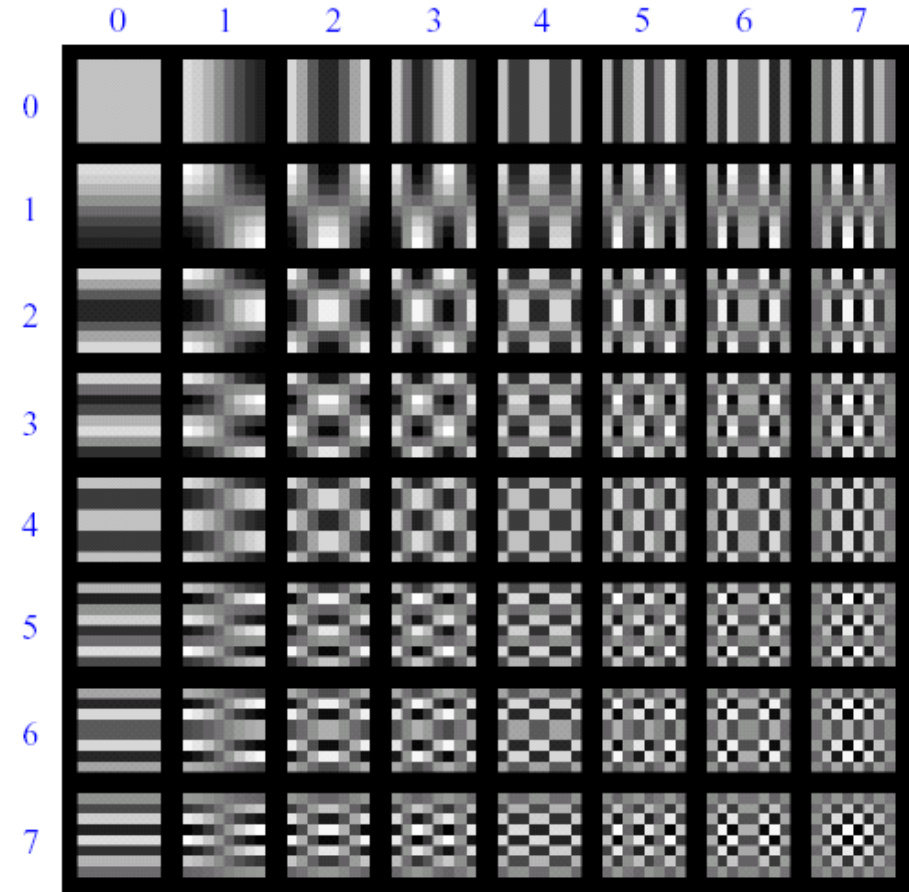
Background: Change of Basis

For vectors and for image patches

Related concept: Image Compression

How is it that a 4MP image can be compressed to a few hundred KB without a noticeable change?

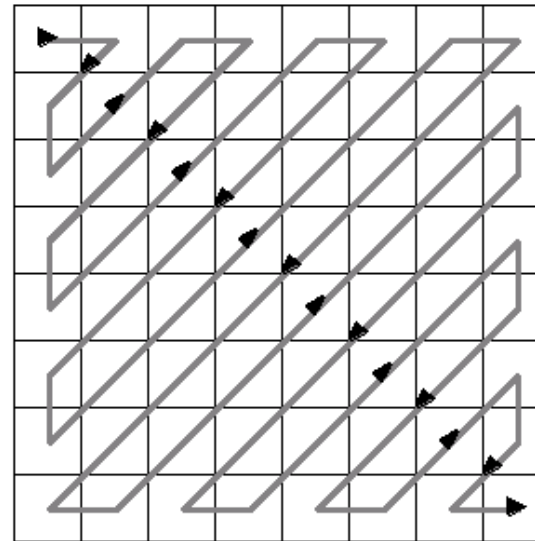
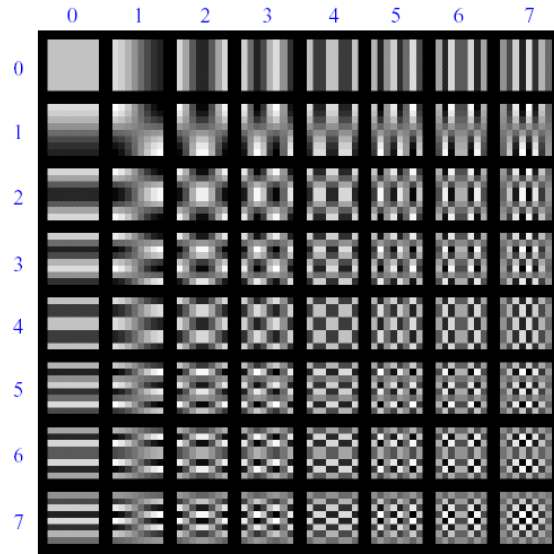
Lossy Image Compression (JPEG)



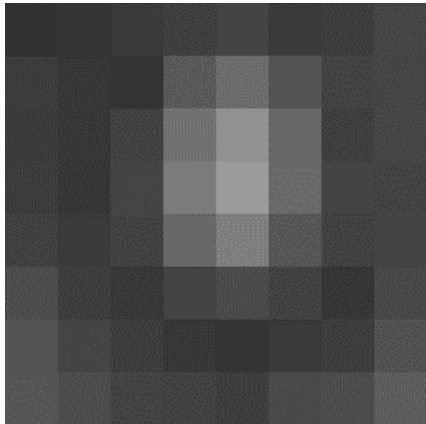
Block-based Discrete Cosine Transform (DCT)

Using DCT in JPEG

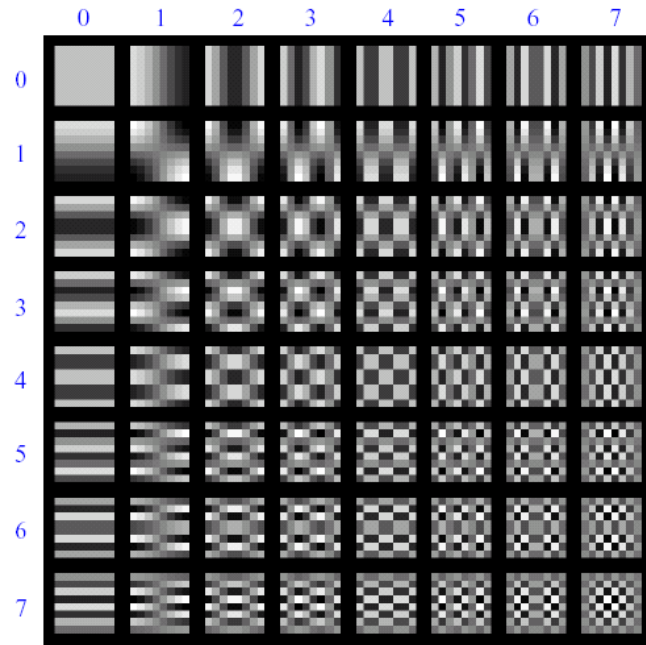
- The first coefficient $B(0,0)$ is the DC component, the average intensity
- The top-left coeffs represent low frequencies, the bottom right – high frequencies



Lossy Image Compression (JPEG)



8x8 image patch



DCT bases

$$G = \begin{matrix} & & & & \xrightarrow{u} & & & & \\ \begin{matrix} \downarrow v \\ \end{matrix} & \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.13 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.88 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix} & \end{matrix}$$

Patch representation after projecting on to DCT bases

Image compression using DCT

- Quantize
 - More coarsely for high frequencies (which also tend to have smaller values)
 - Many quantized high frequency values will be zero
- Encode
 - Can decode with inverse dct

Filter responses

$$G = \begin{matrix} & & & \xrightarrow{u} & & & & & \\ \begin{matrix} \left[\begin{array}{cccccccc} -415.38 & -30.19 & -61.20 & 27.24 & 56.13 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.88 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{array} \right] \end{matrix} & & \downarrow v \end{matrix}$$

Quantized values

$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Quantization table

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

JPEG Compression Summary

1. Convert image to YCrCb
2. Subsample color by factor of 2
 - People have bad resolution for color
3. Split into blocks (8x8, typically), subtract 128
4. For each block
 - a. Compute DCT coefficients
 - b. Coarsely quantize
 - Many high frequency components will become zero
 - c. Encode (e.g., with Huffman coding)

<http://en.wikipedia.org/wiki/YCbCr>

<http://en.wikipedia.org/wiki/JPEG>

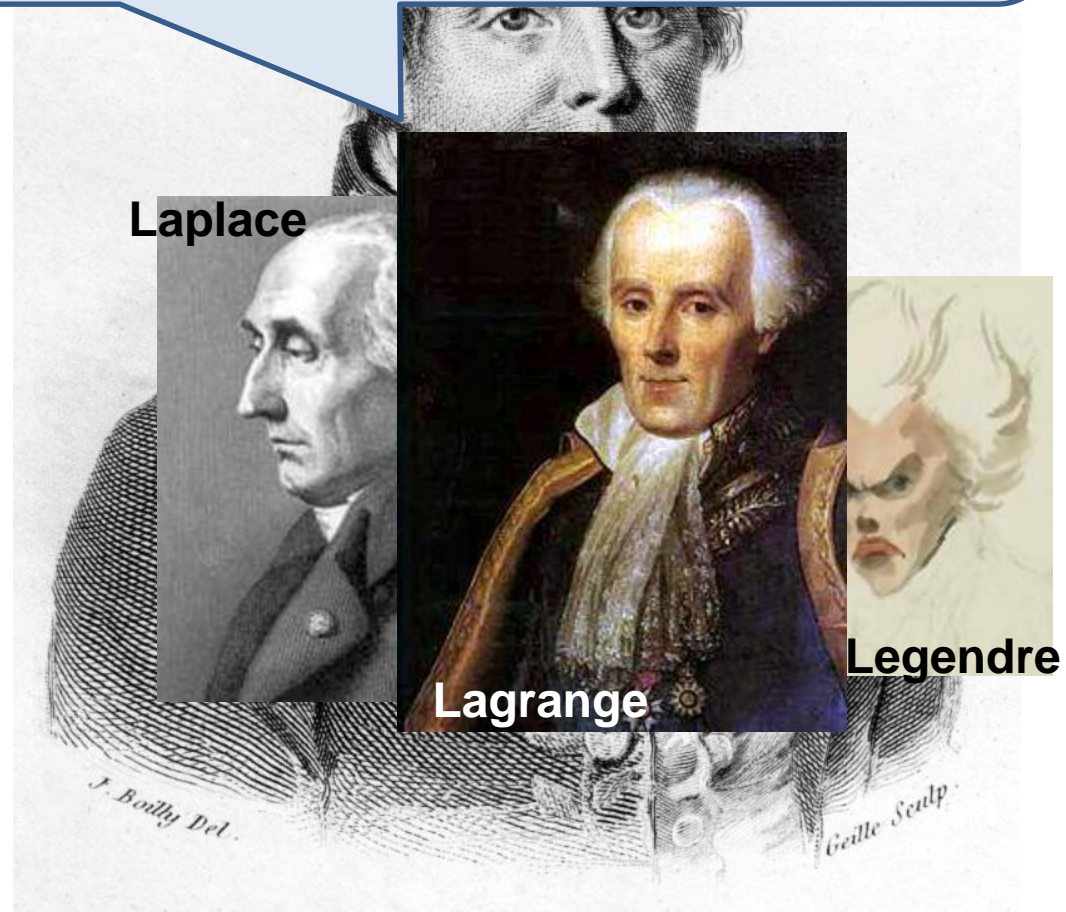
Jean Baptiste Joseph Fourier (1768-1830)

had crazy idea (1807):

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

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

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



Fourier, Joseph (1768-1830)



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](#)

Additional biographies: [MacTutor \(St. Andrews\)](#), [Bonn](#)

© 1996-2007 Eric W. Weisstein

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

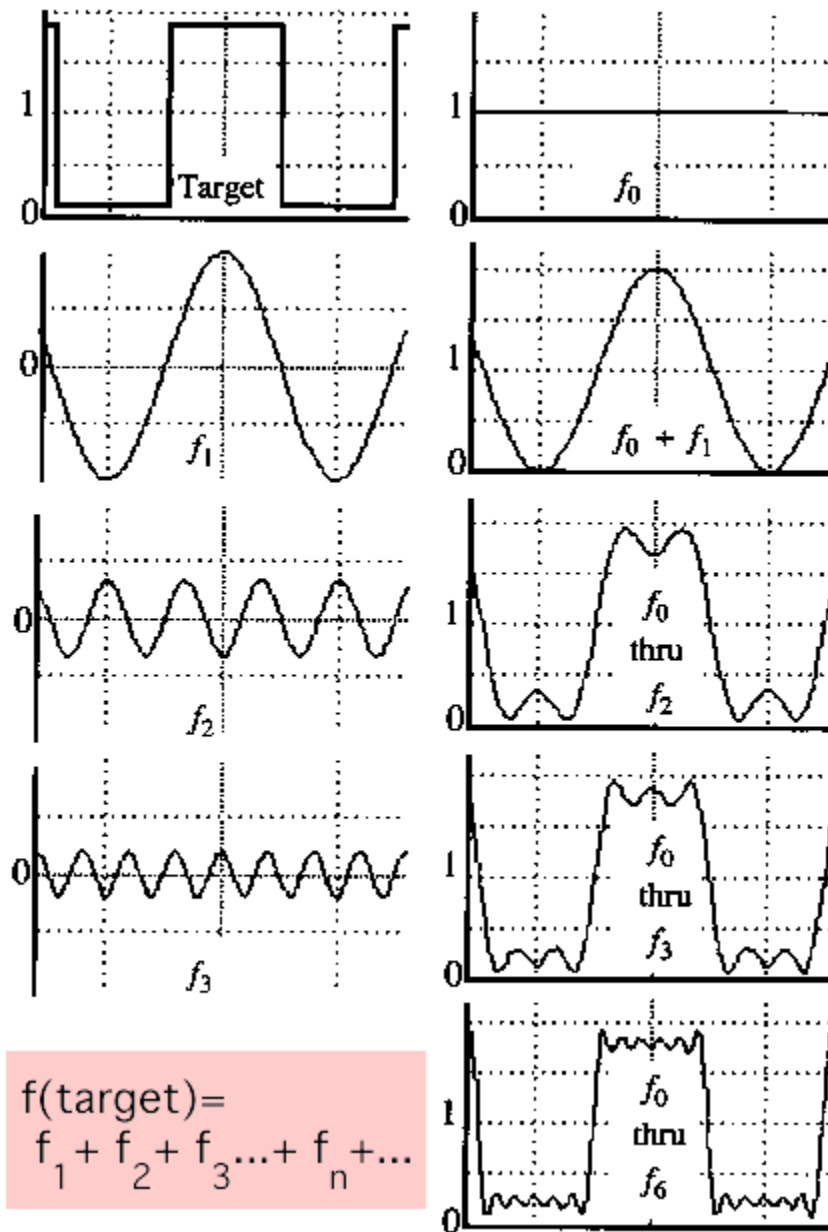


A sum of sines

Our building block:

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

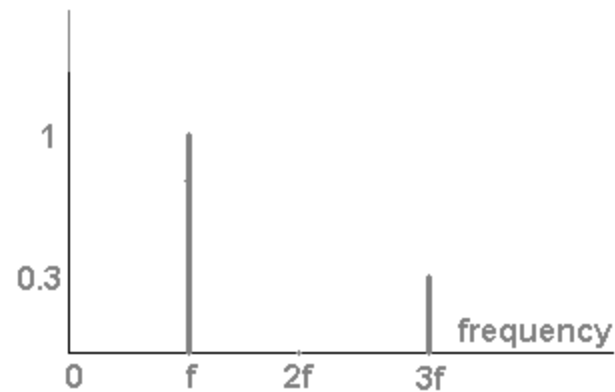
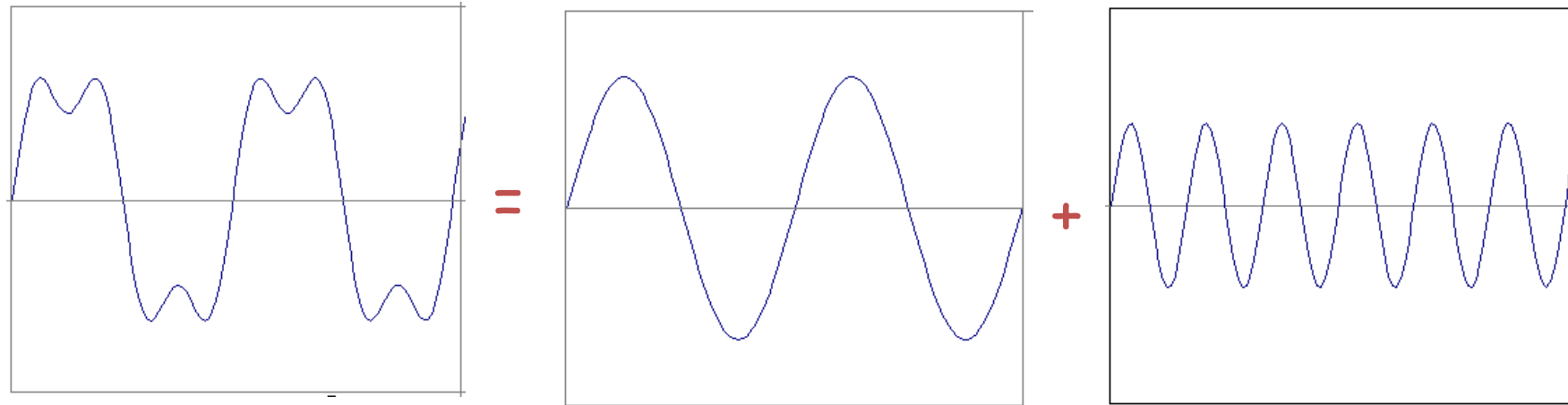
Add enough of them to get any signal $g(x)$ you want!



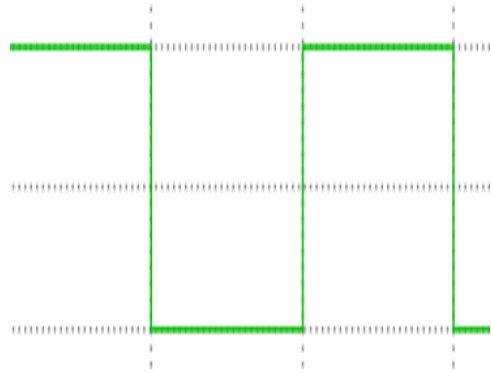
$$f(\text{target}) = f_1 + f_2 + f_3 + \dots + f_n + \dots$$

Frequency Spectra

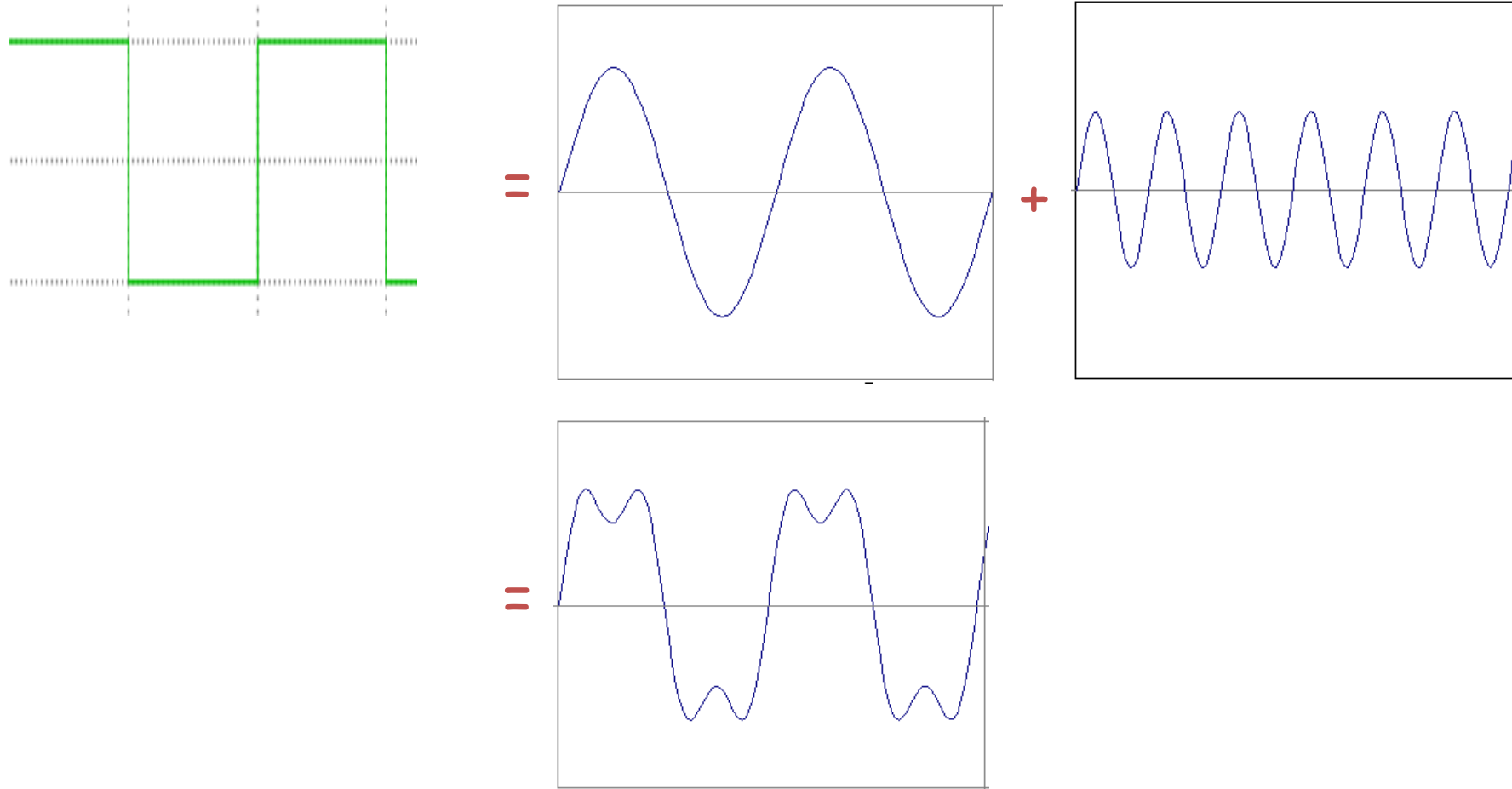
- example : $g(t) = \sin(2\pi f t) + (1/3)\sin(2\pi(3f) t)$



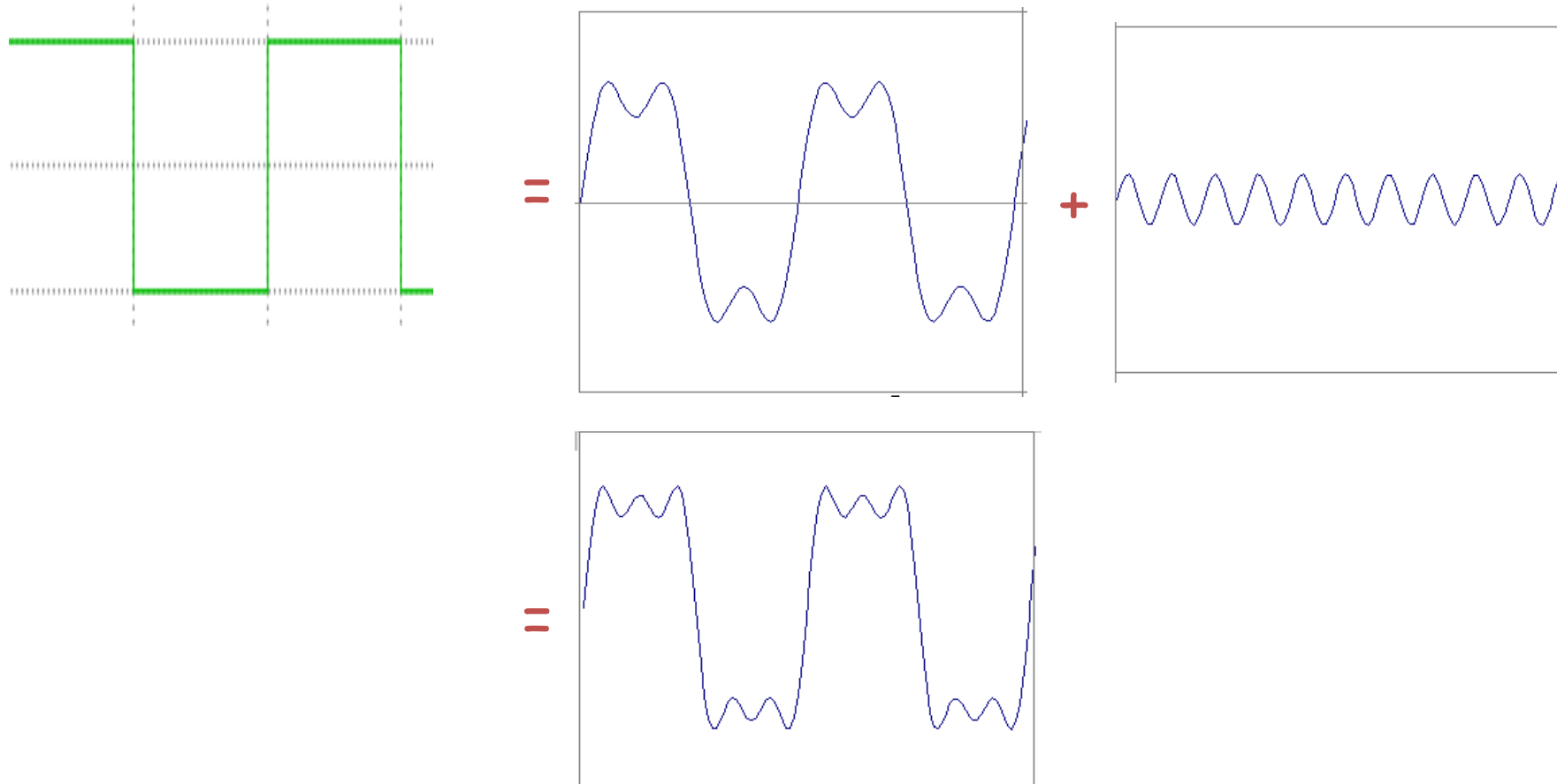
Frequency Spectra



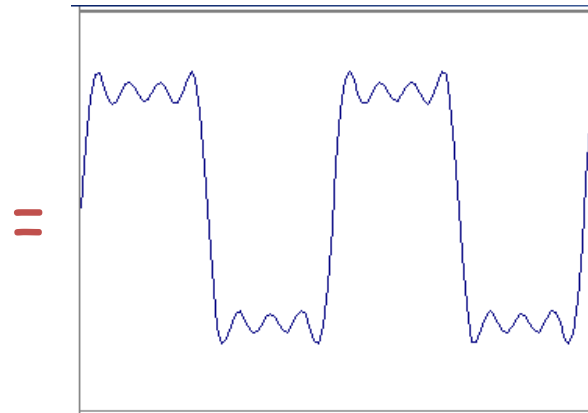
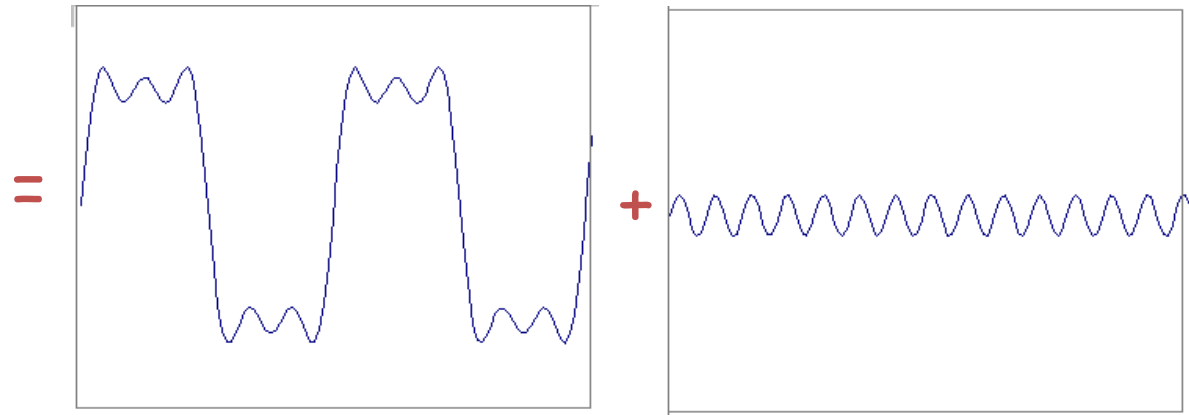
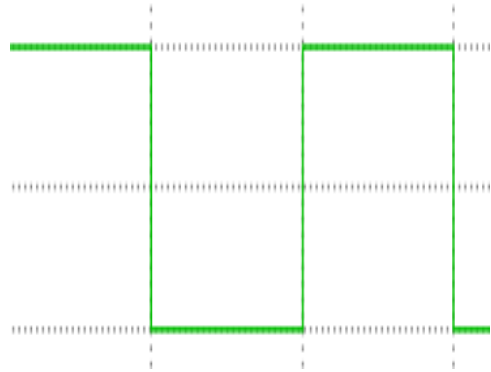
Frequency Spectra



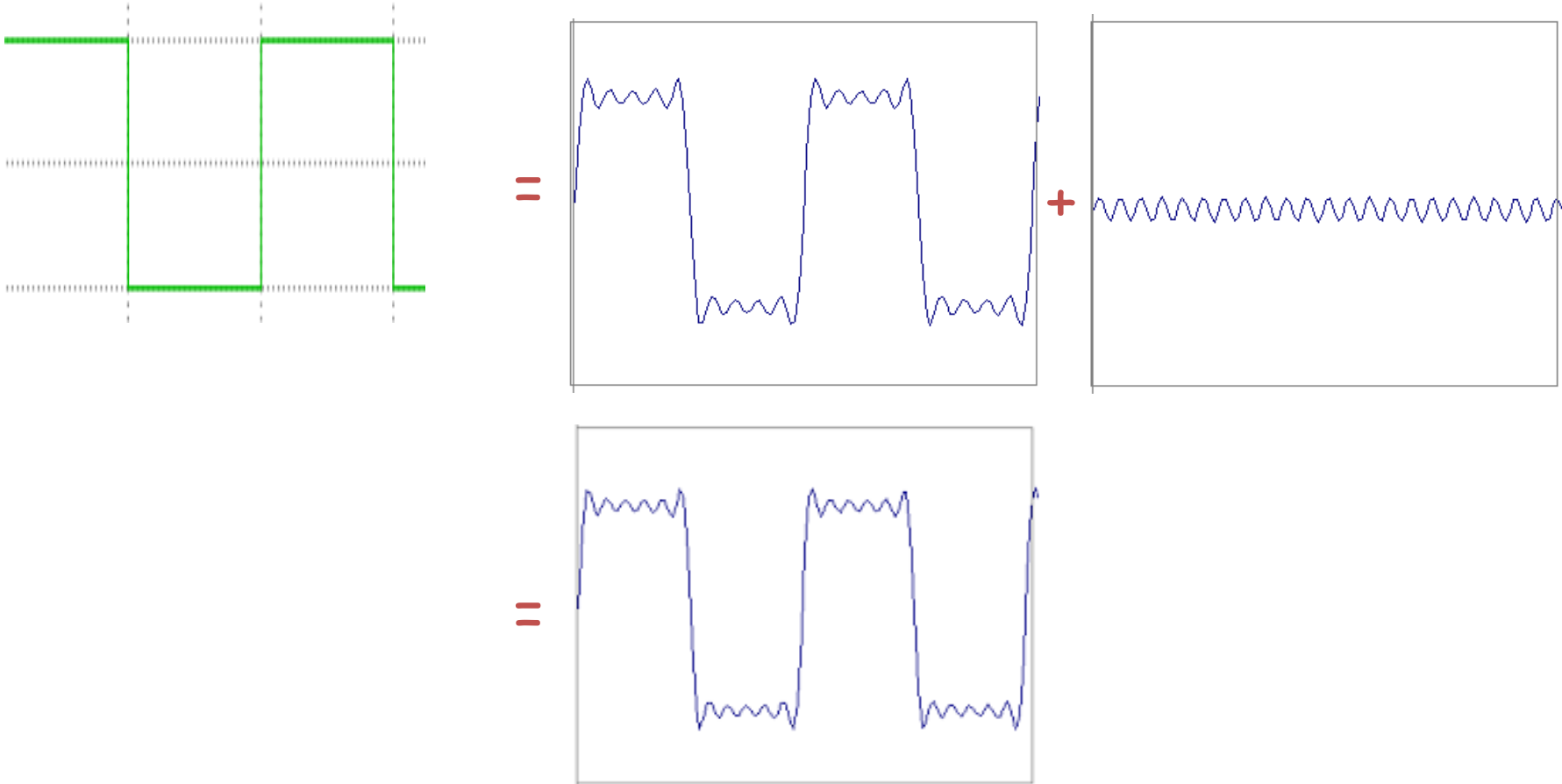
Frequency Spectra



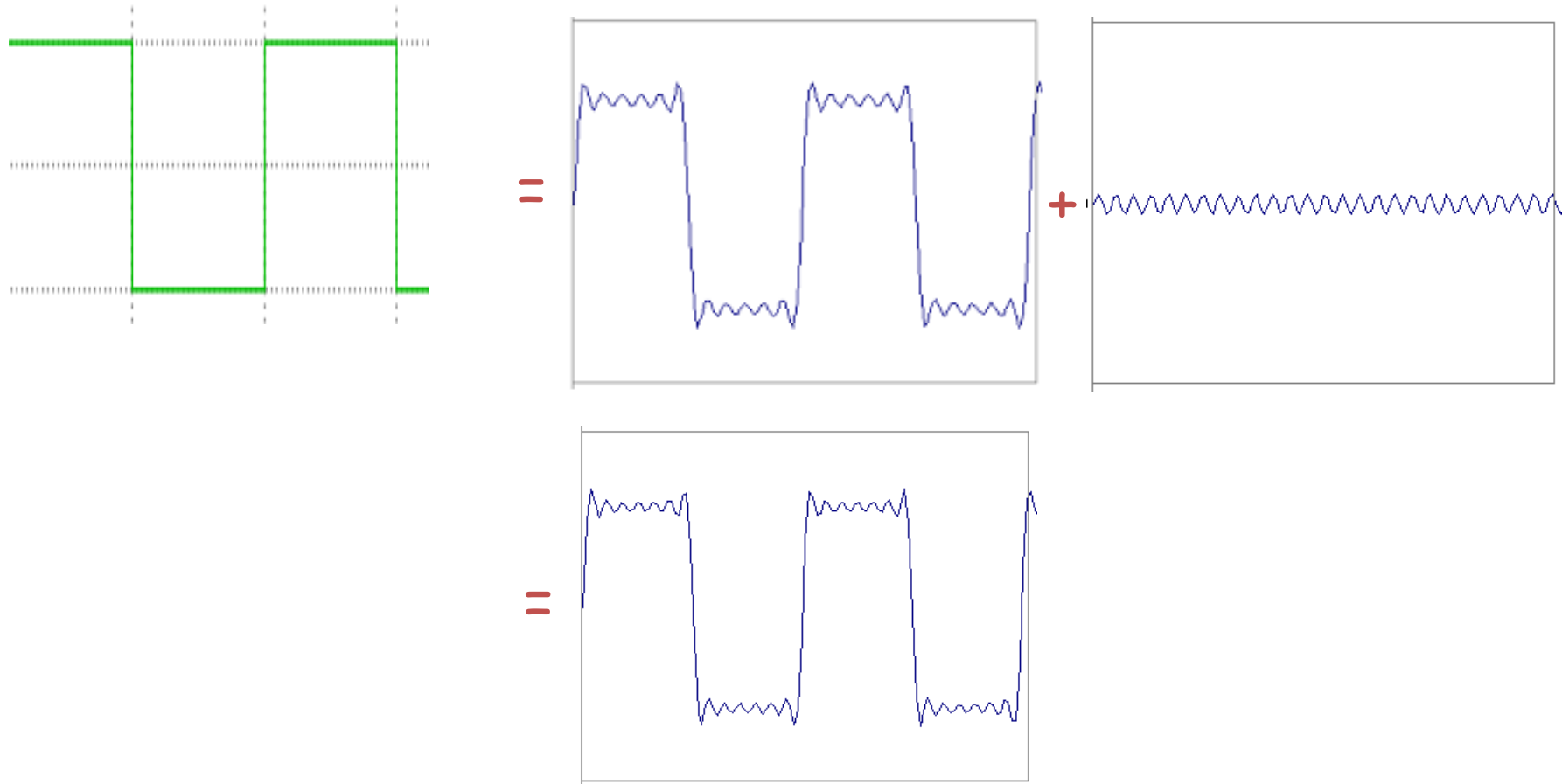
Frequency Spectra



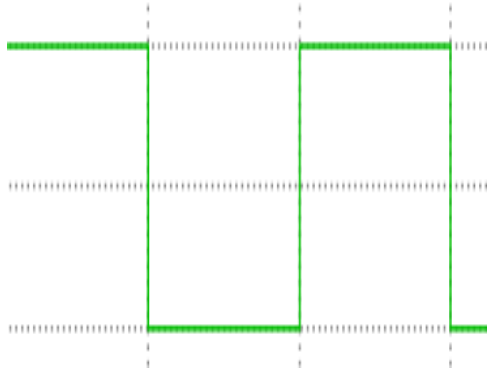
Frequency Spectra



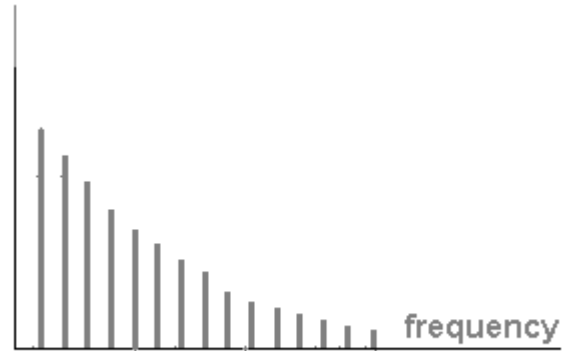
Frequency Spectra



Frequency Spectra

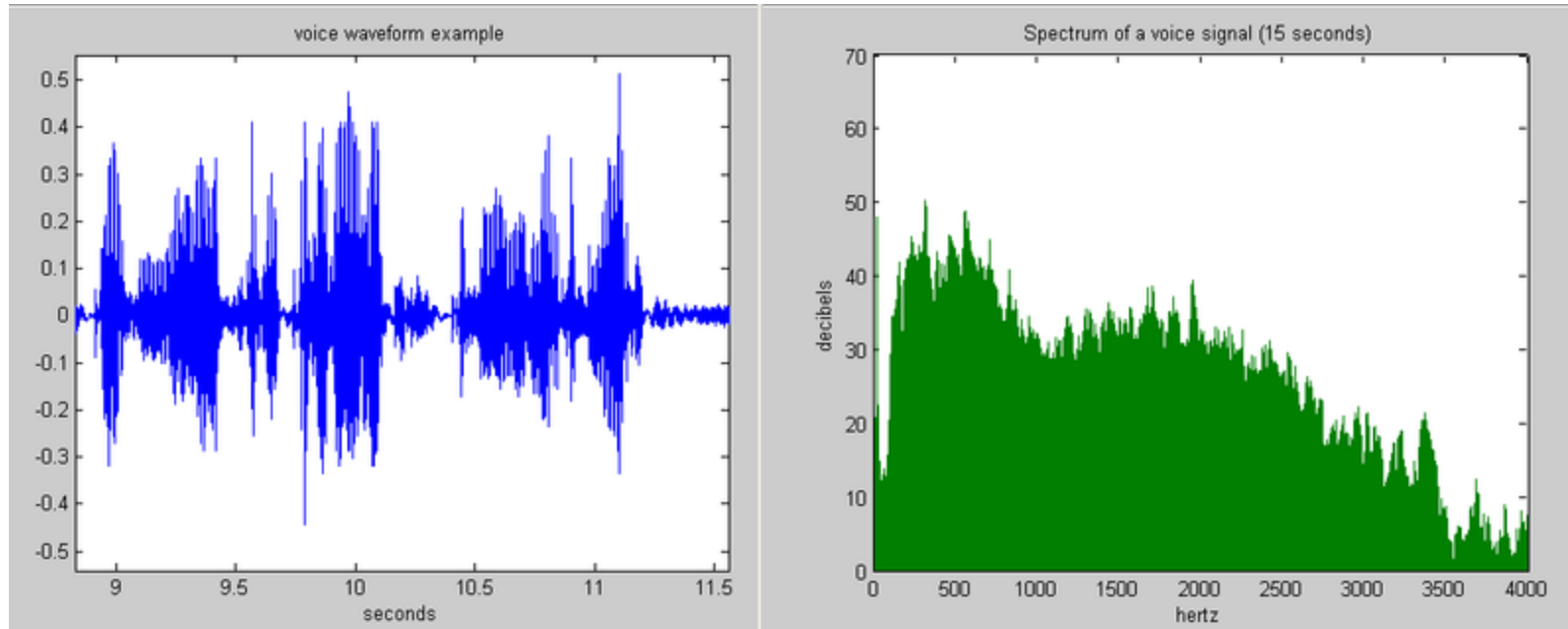


$$= A \sum_{k=1}^{\infty} \frac{1}{k} \sin(2\pi kt)$$



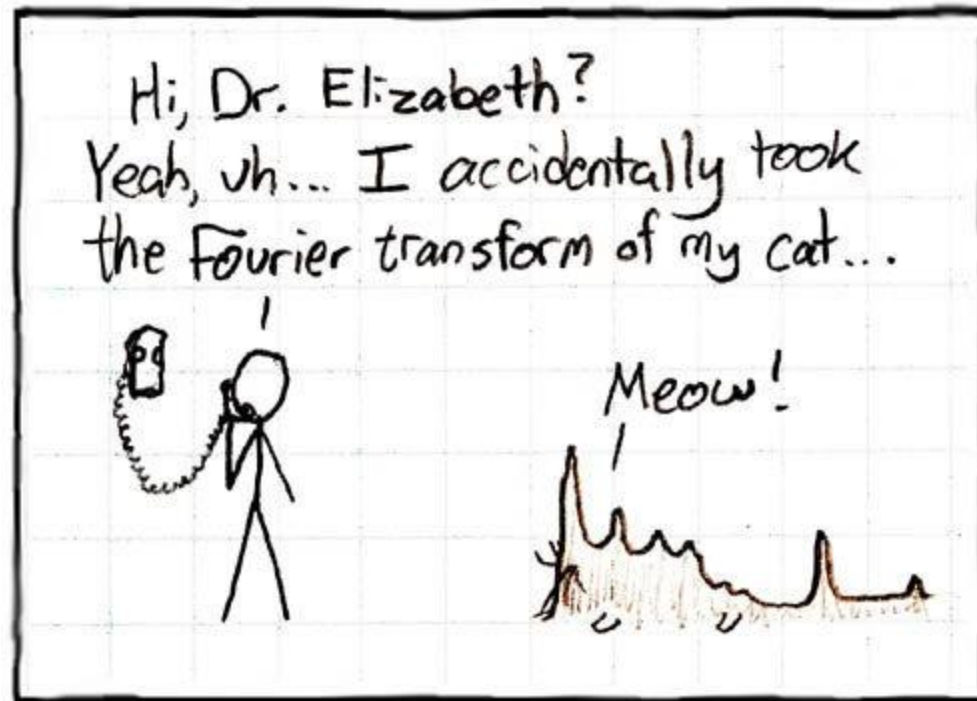
Example: Music

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



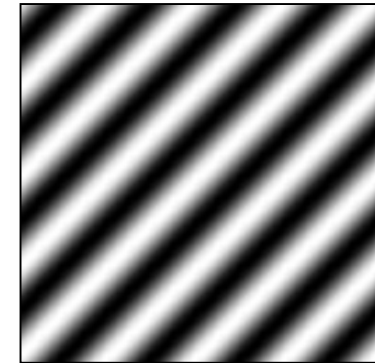
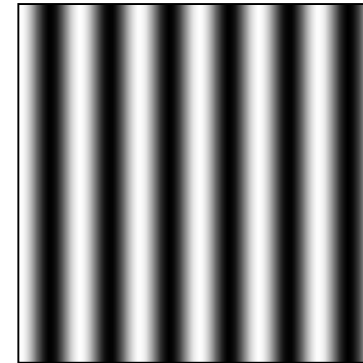
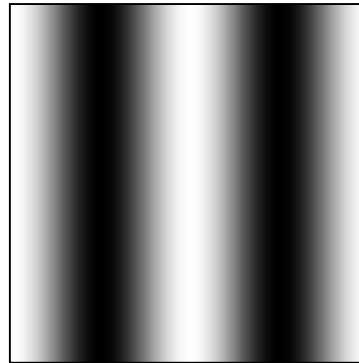
Other signals

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

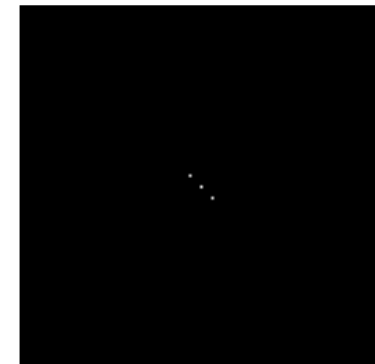
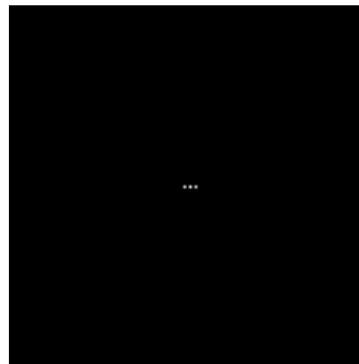


Fourier analysis in images

Intensity Image



Fourier Image



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

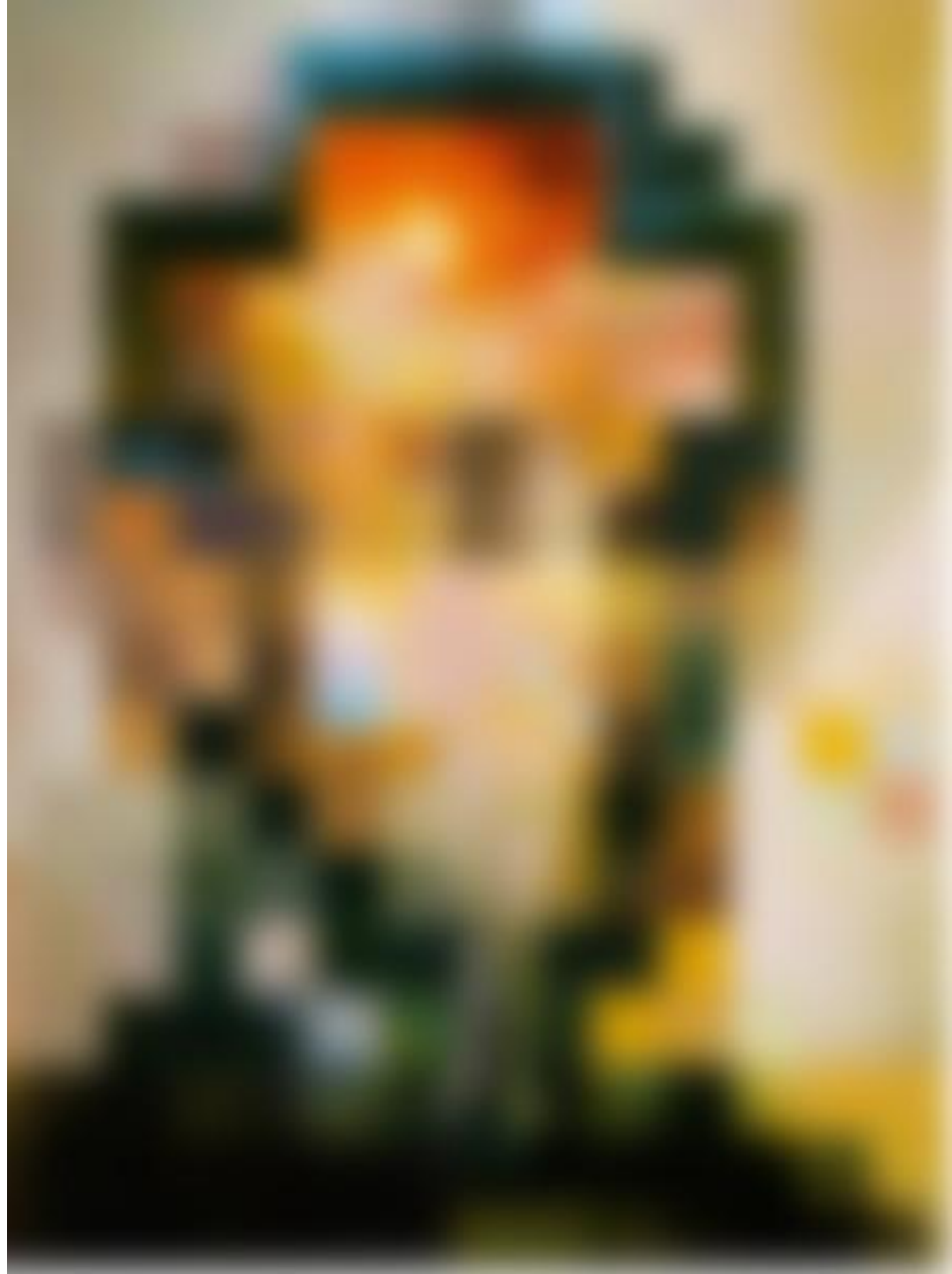
Amplitude: $A = \pm\sqrt{R(\omega)^2 + I(\omega)^2}$

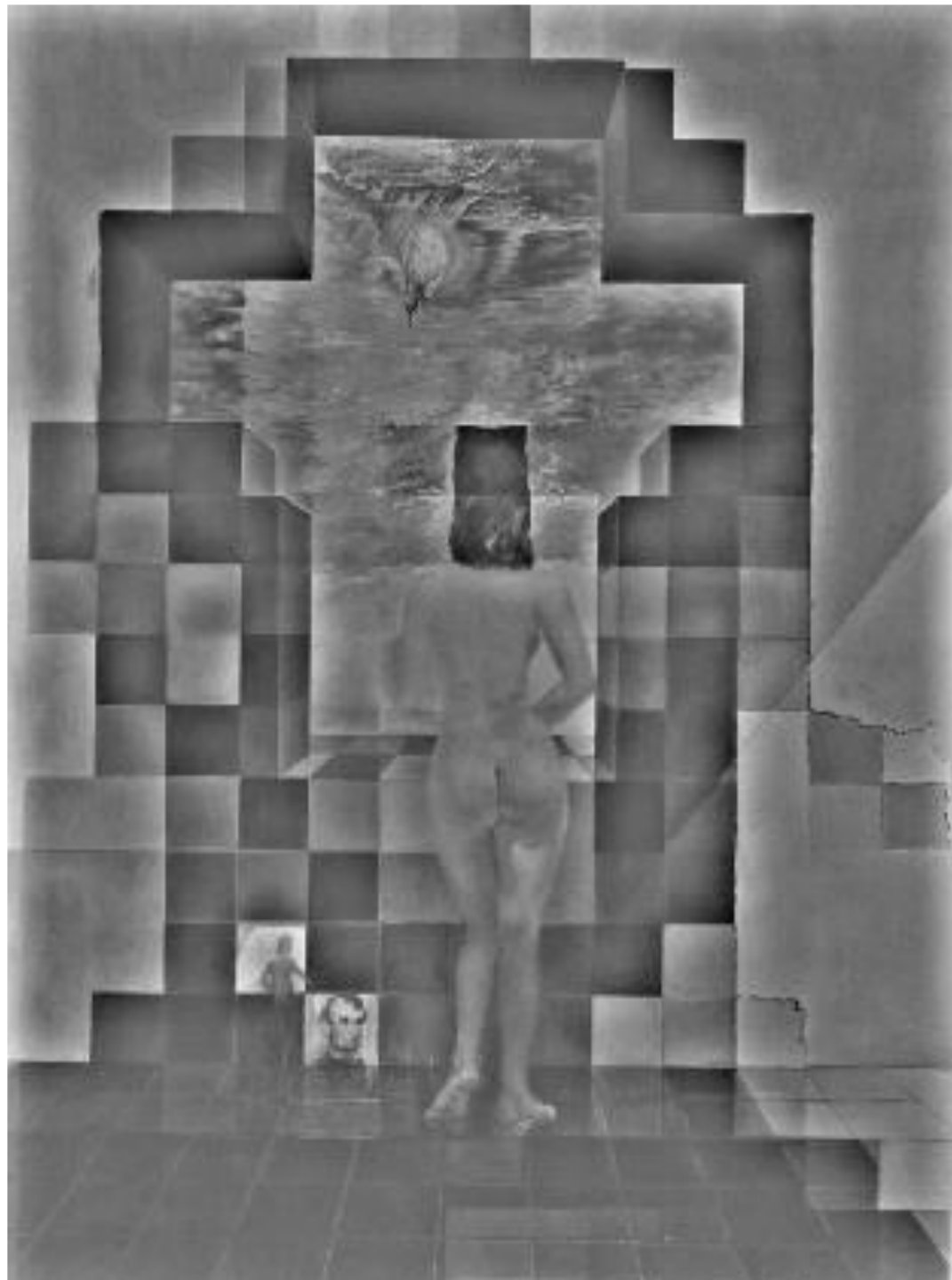
Phase: $\phi = \tan^{-1} \frac{I(\omega)}{R(\omega)}$

Salvador Dali invented Hybrid Images?

Salvador Dali
*"Gala Contemplating the Mediterranean Sea,
which at 20 meters becomes the portrait
of Abraham Lincoln", 1976*

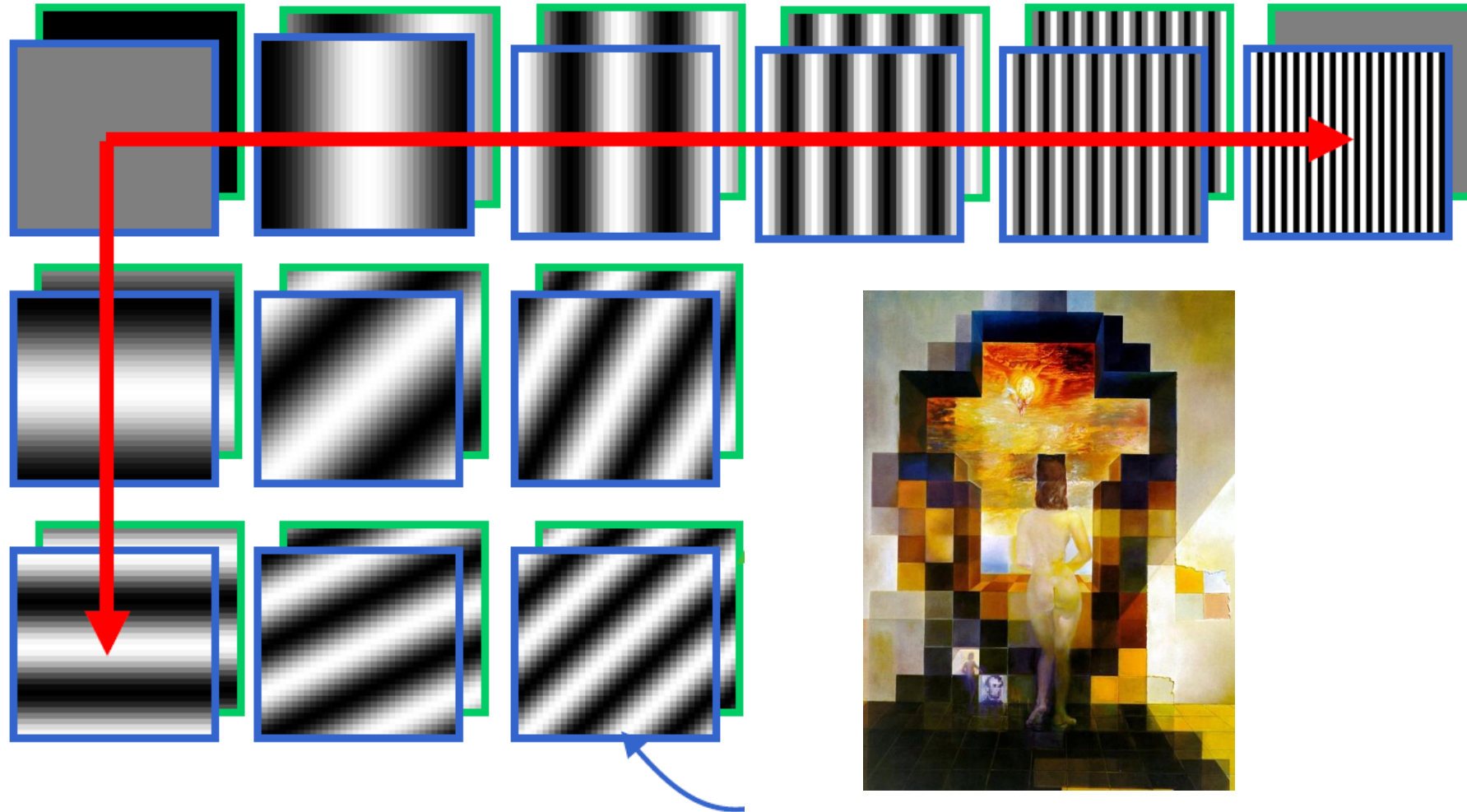






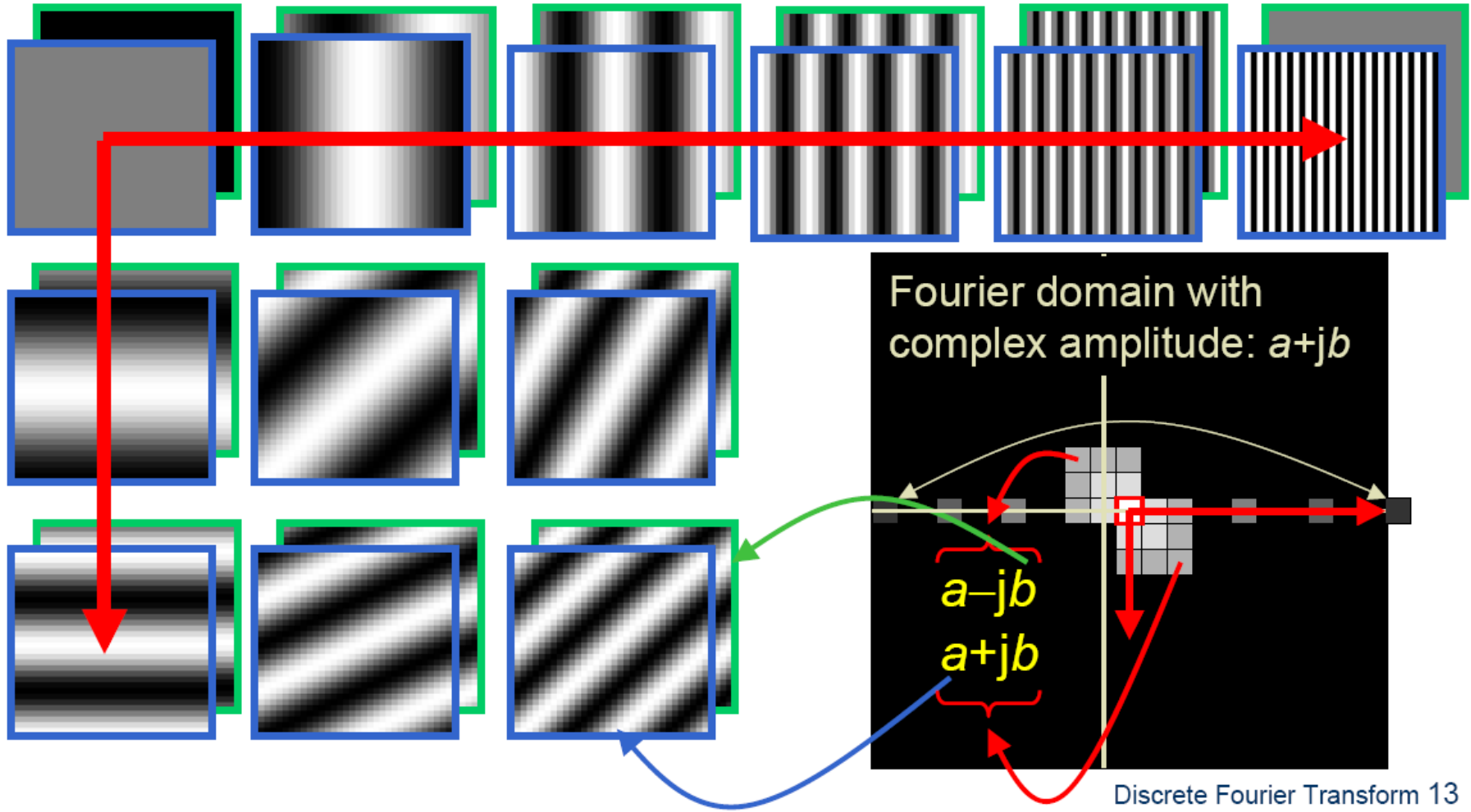
Fourier Bases

Teases away fast vs. slow changes in the image.

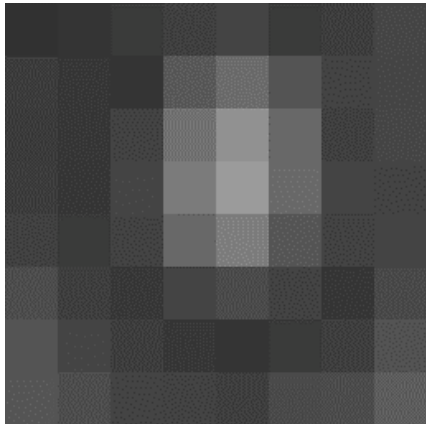


This change of basis is the Fourier Transform

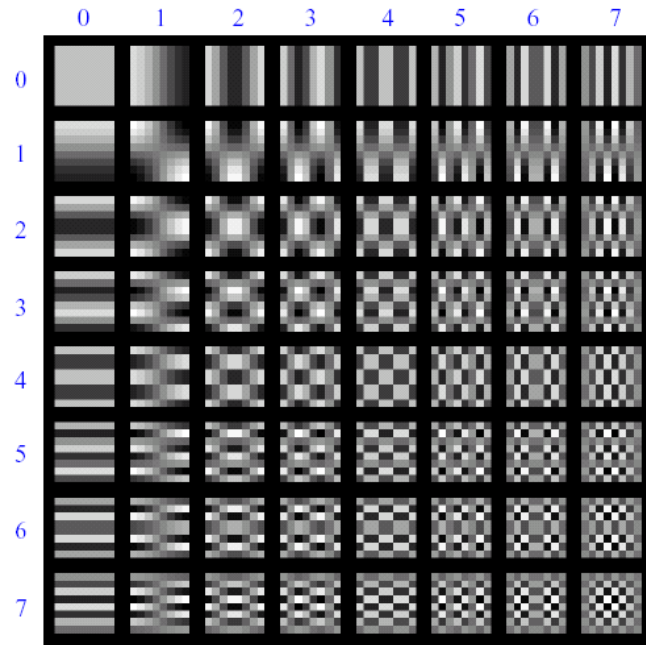
Fourier Bases



This looks a lot like DCT in JPEG compression



8x8 image patch

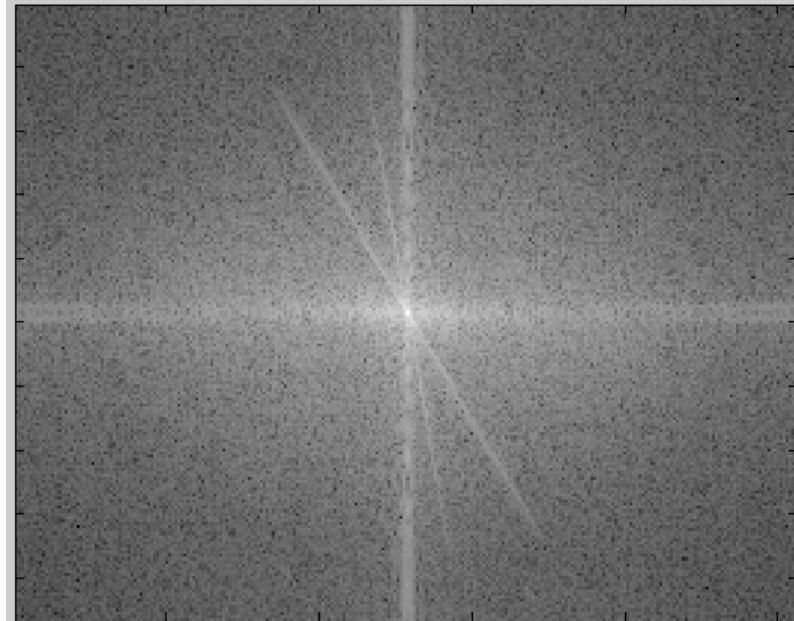


DCT bases

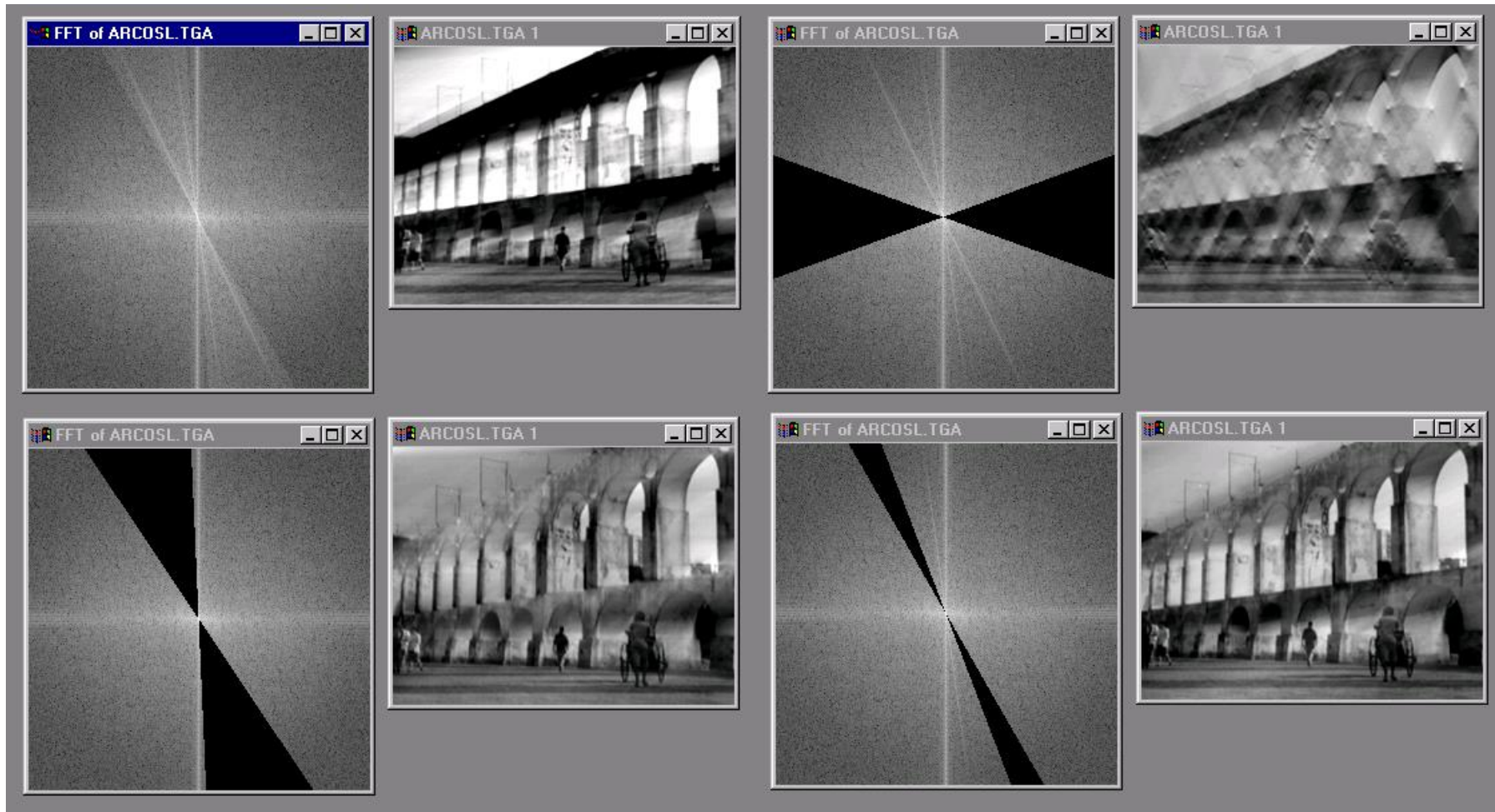
$$G = \begin{matrix} & & & & \xrightarrow{u} & & & & \\ \begin{matrix} \downarrow v \\ \end{matrix} & \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.13 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.88 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix} & \end{matrix}$$

Patch representation after projecting on to DCT bases

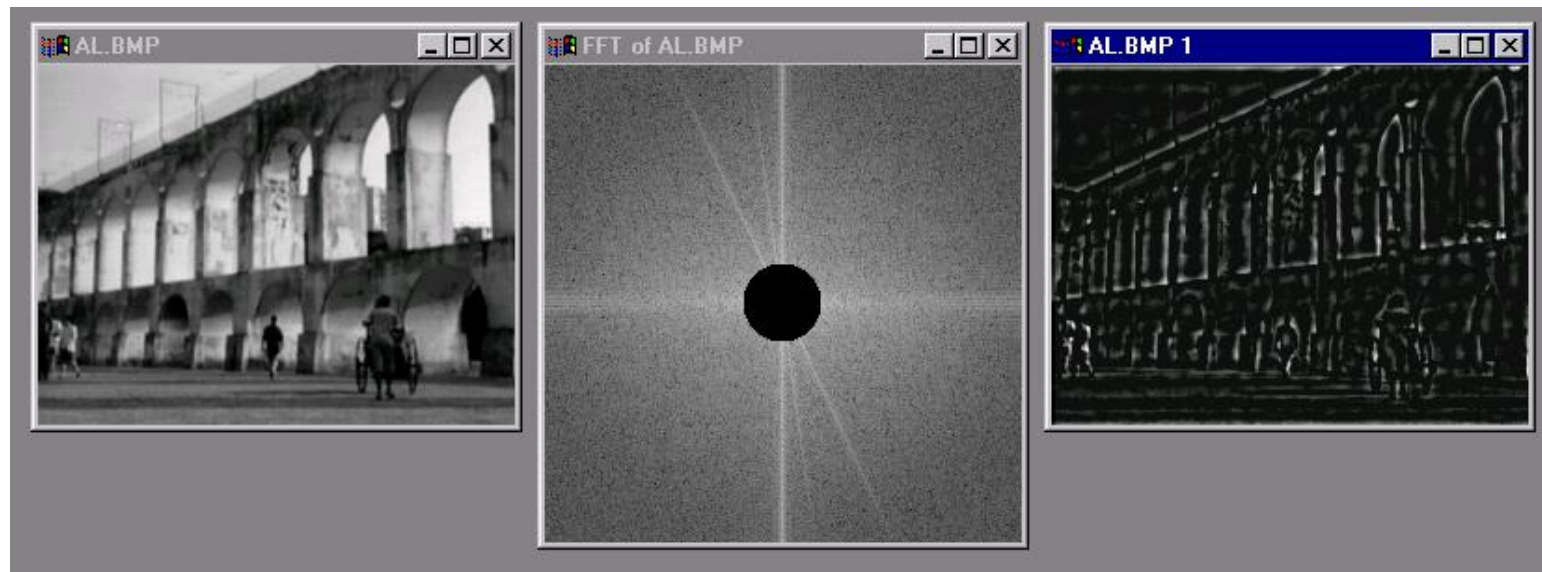
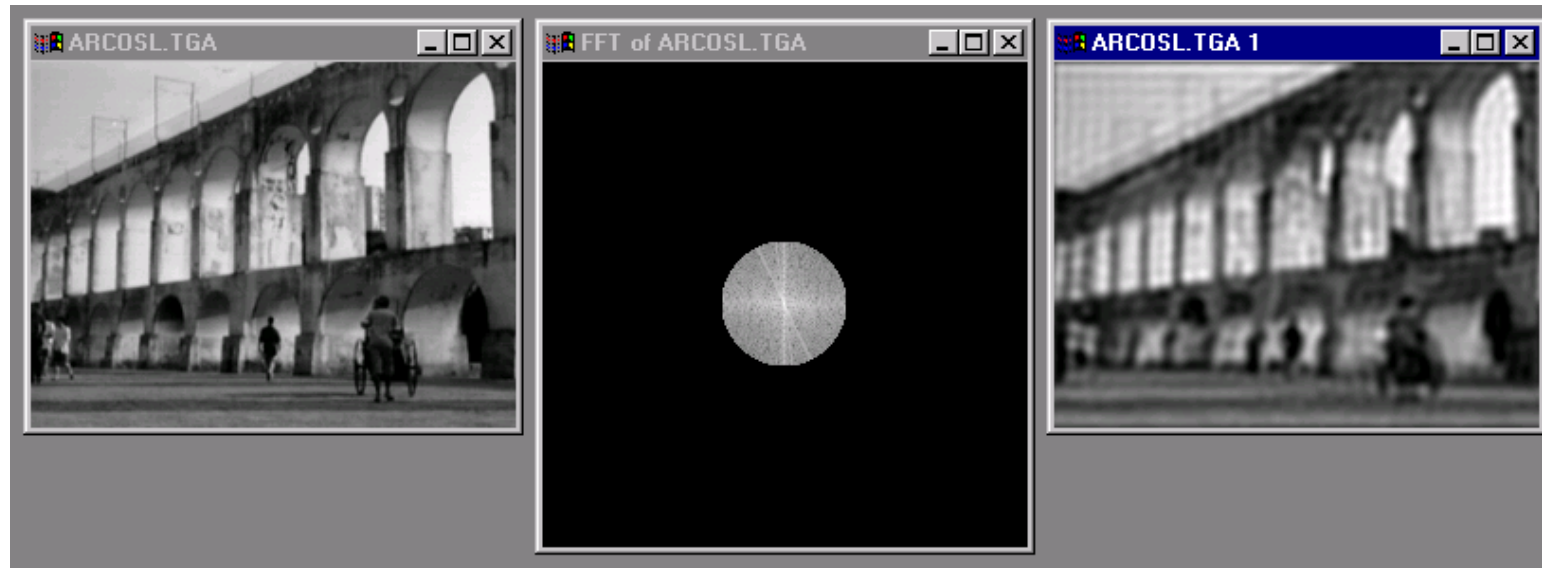
Man-made Scene



Can change spectrum, then reconstruct



Low and High Pass filtering



Computing the Fourier Transform

$$H(\omega) = \mathcal{F} \{h(x)\} = Ae^{j\phi}$$

Continuous

$$H(\omega) = \int_{-\infty}^{\infty} h(x)e^{-j\omega x} dx$$

Discrete

$$H(k) = \frac{1}{N} \sum_{x=0}^{N-1} h(x)e^{-j\frac{2\pi kx}{N}} \quad k = -N/2..N/2$$

Fast Fourier Transform (FFT): $N \log N$

The Convolution Theorem

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

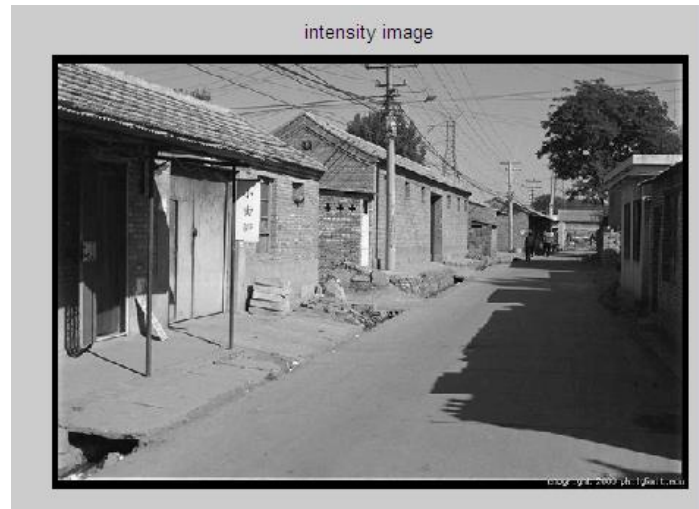
$$F[g * h] = F[g]F[h]$$

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

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

Filtering in spatial domain

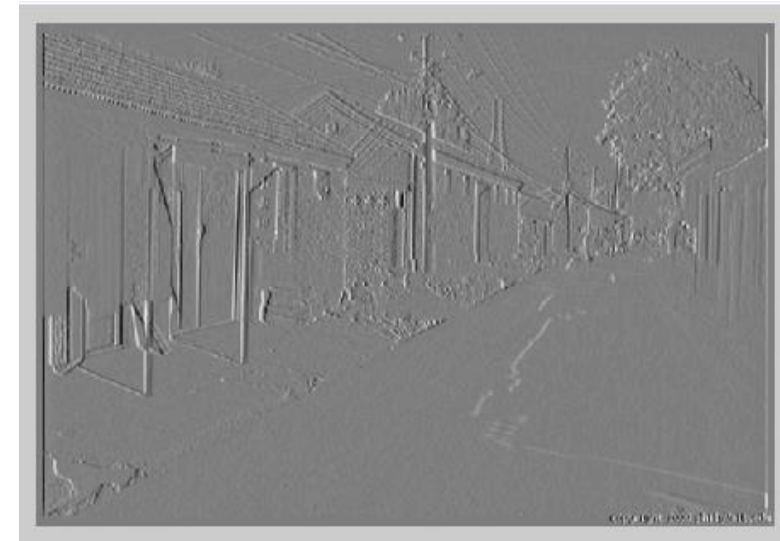
1	0	-1
2	0	-2
1	0	-1



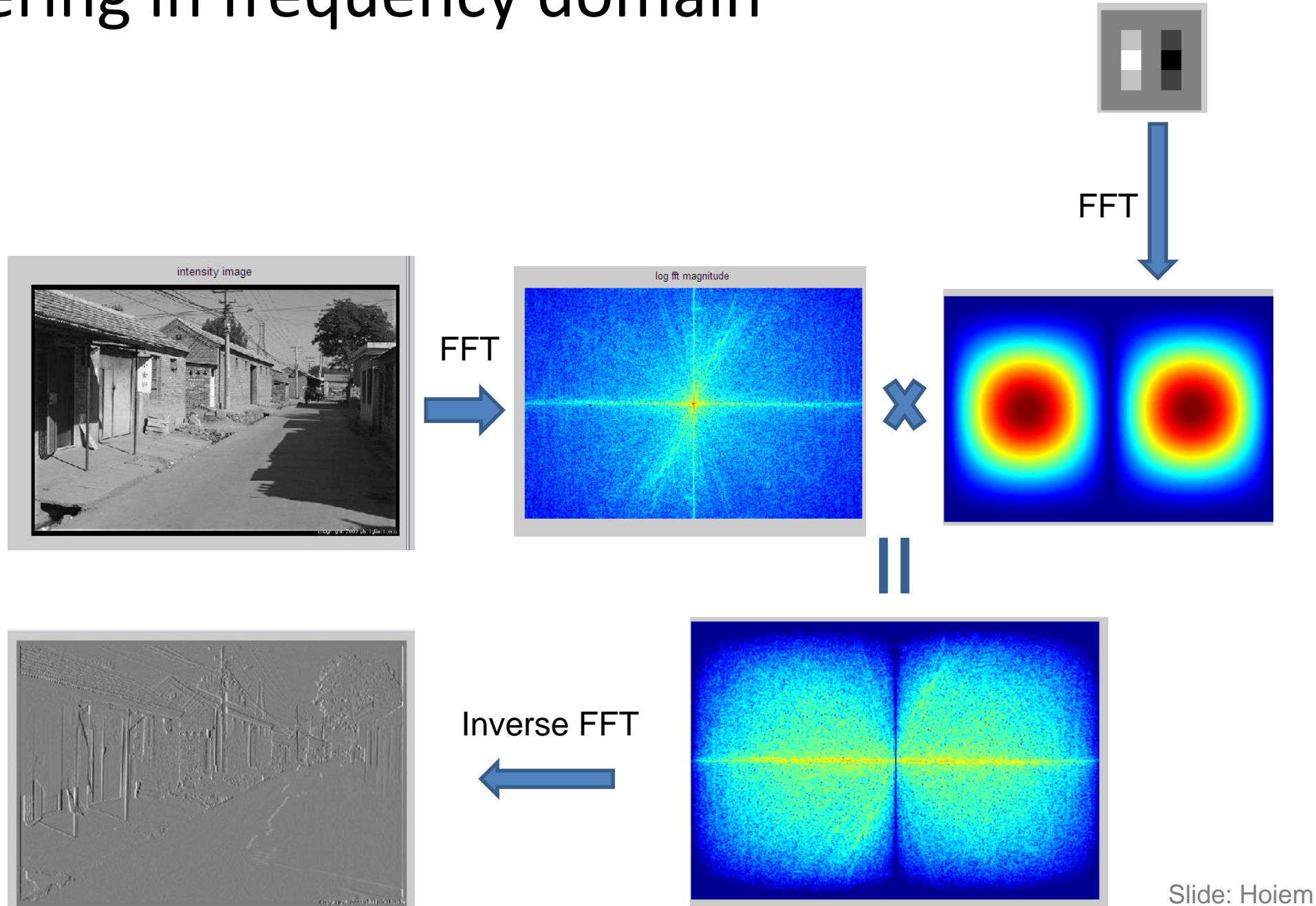
*



=



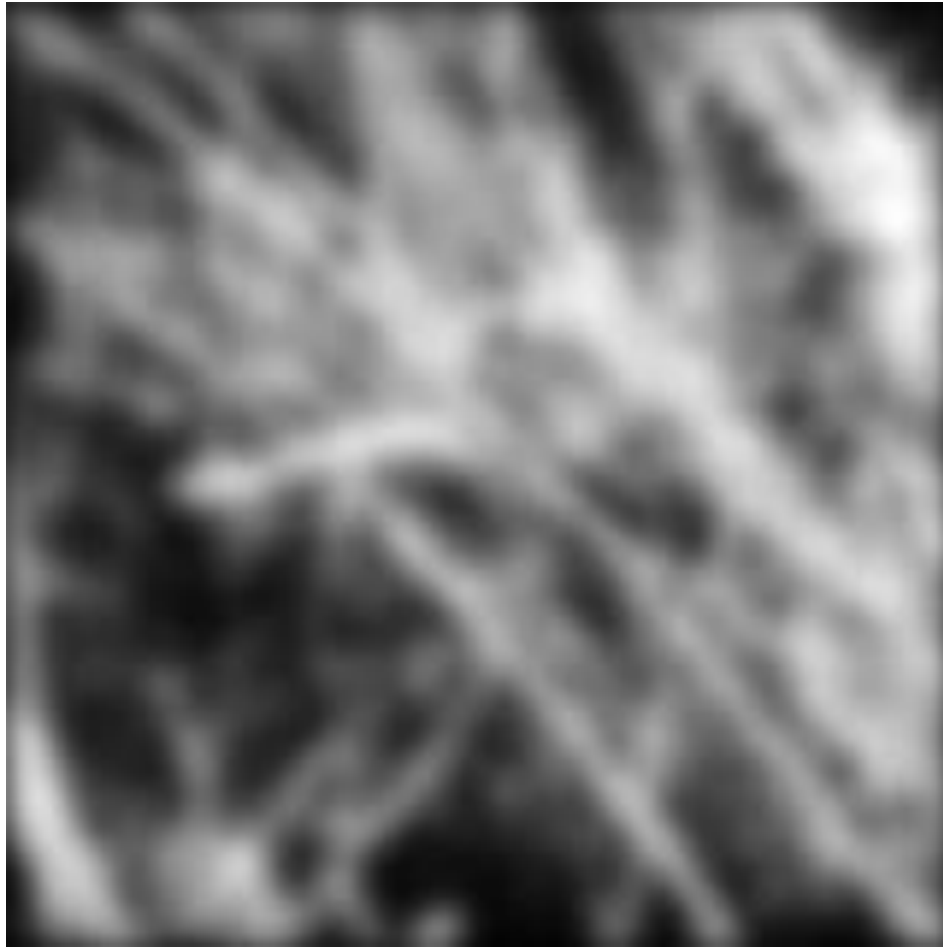
Filtering in frequency domain



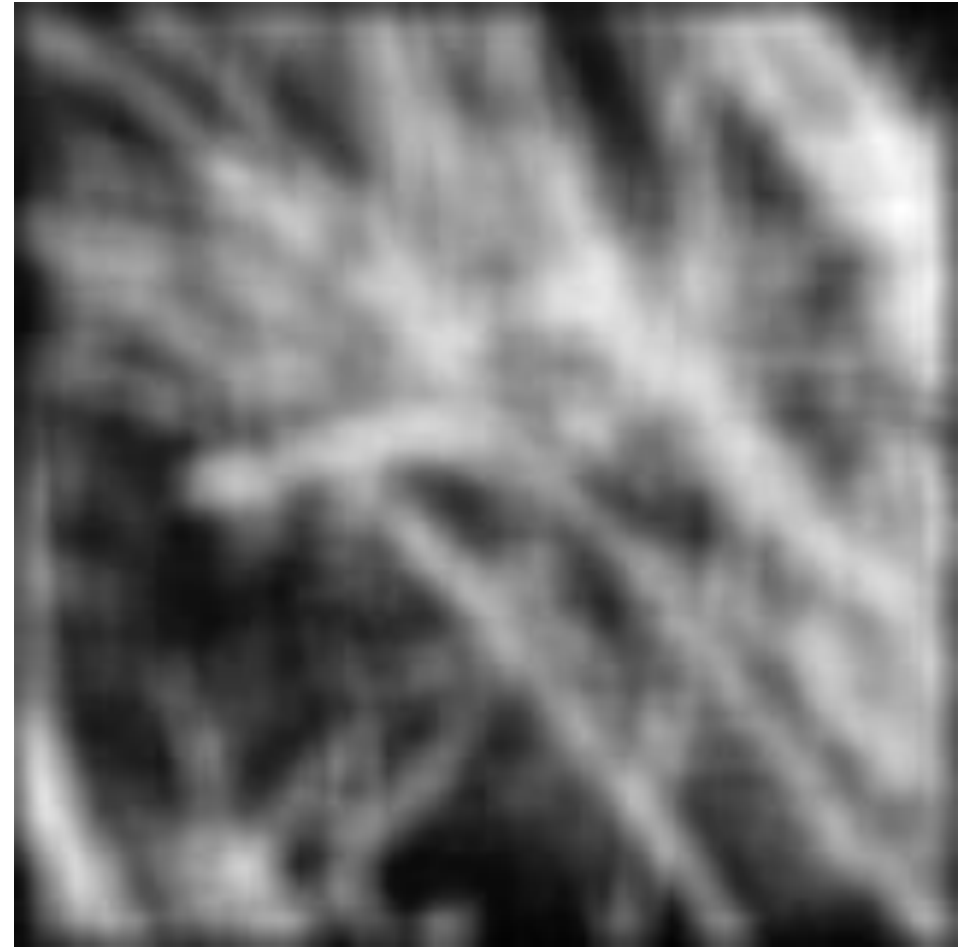
Filtering

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

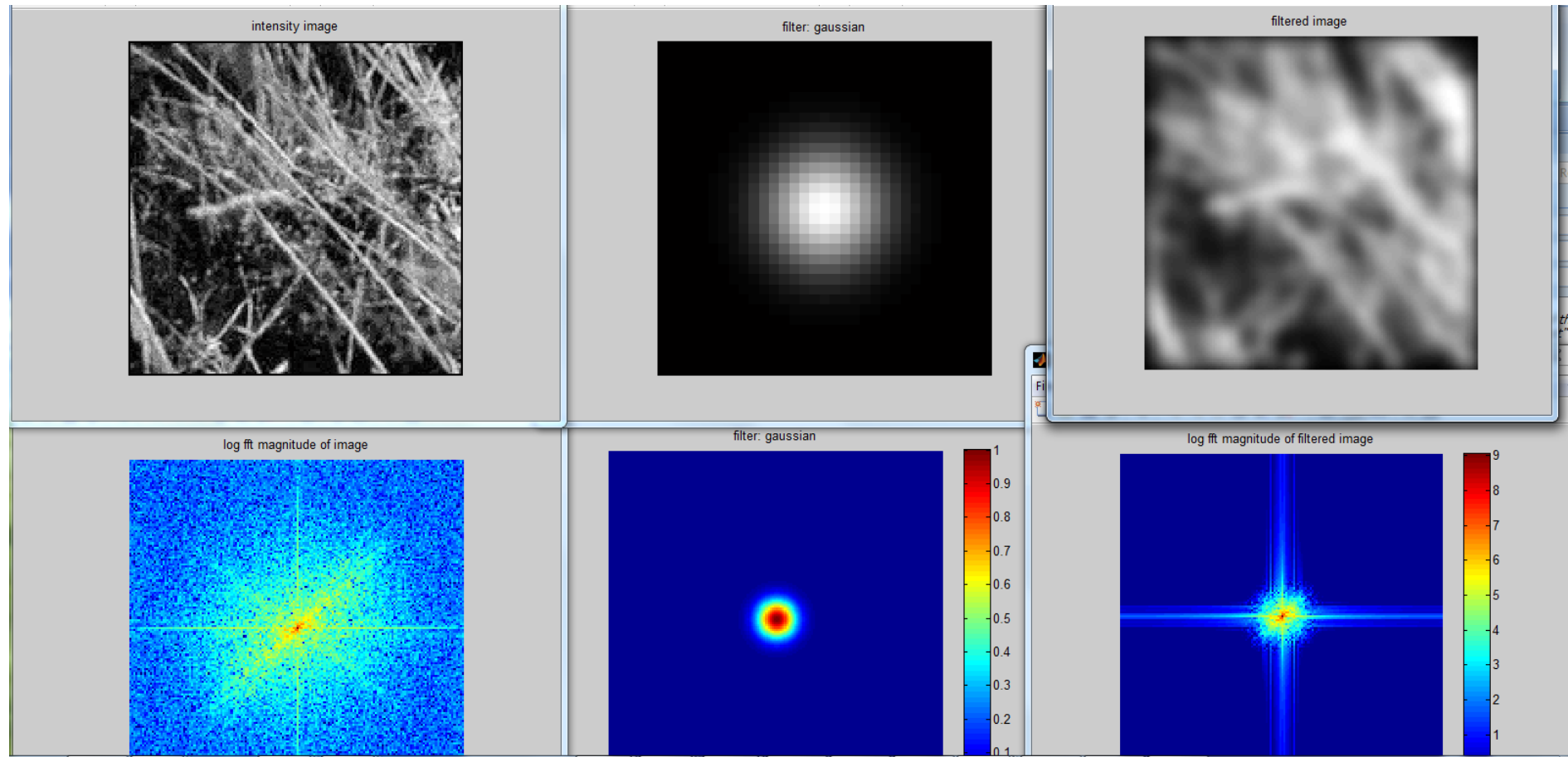
Gaussian



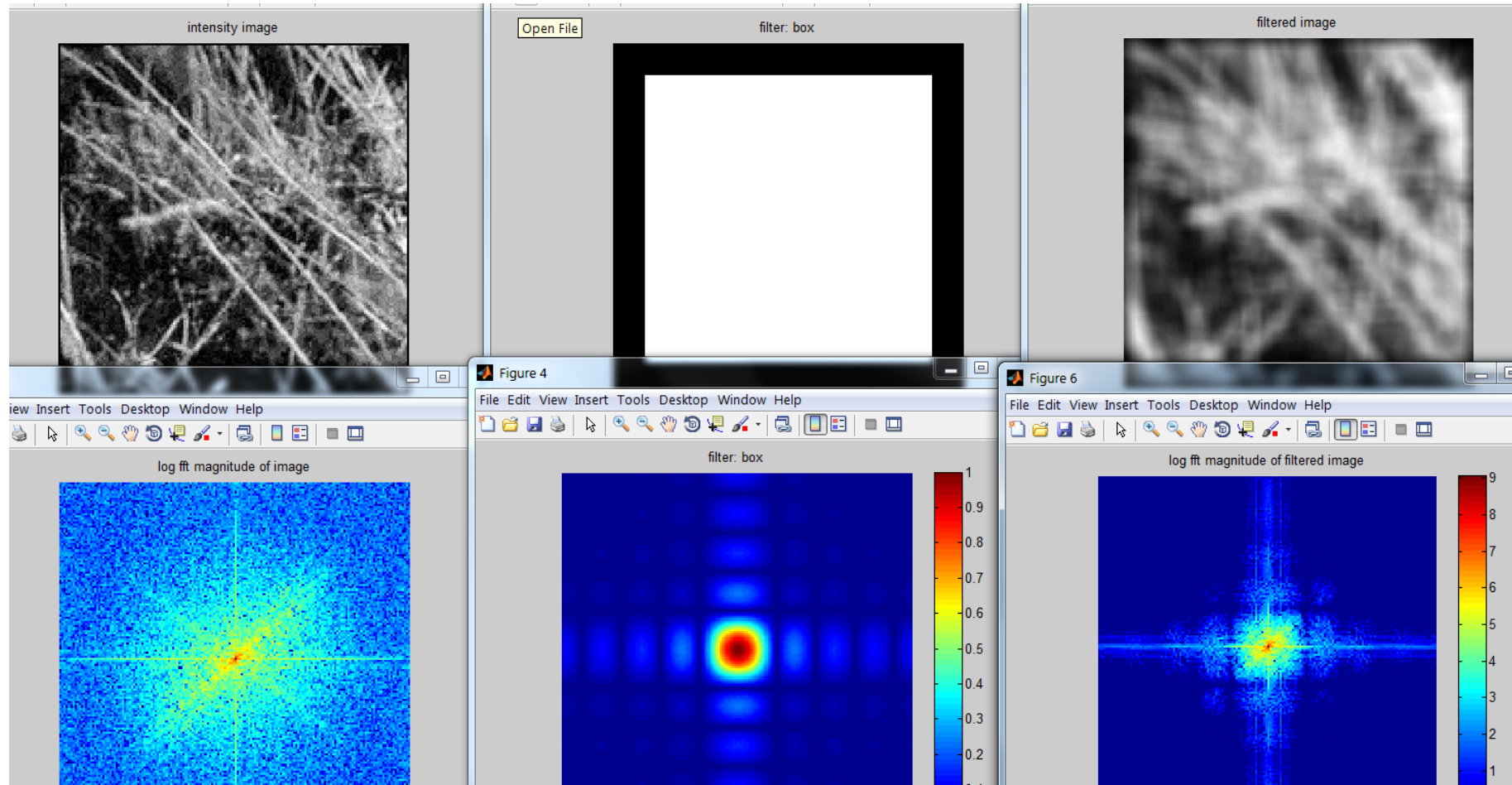
Box filter



Gaussian



Box Filter

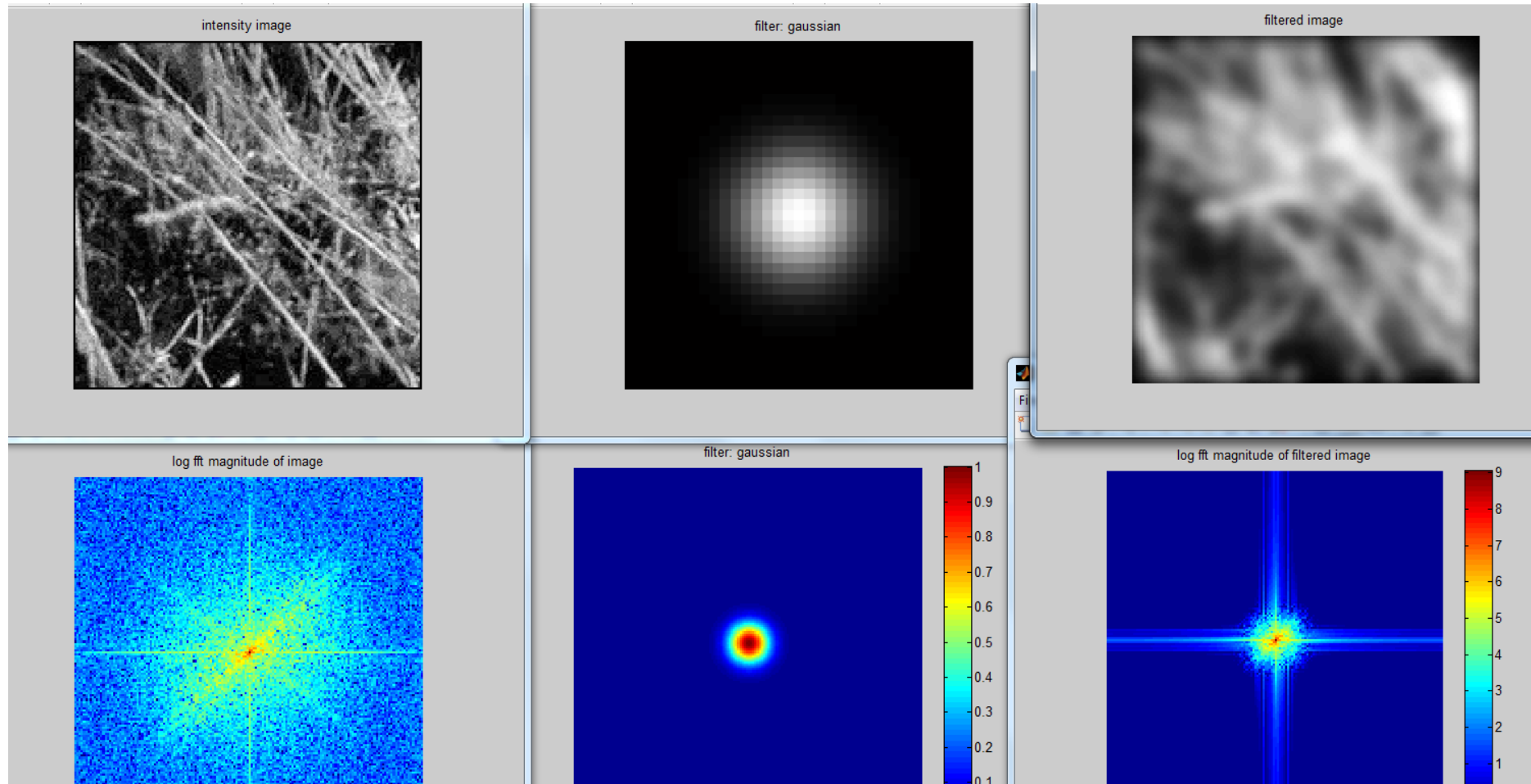


Is convolution invertible?

- If convolution is just multiplication in the Fourier domain, isn't deconvolution just division?
- Sometimes, it clearly is invertible (e.g. a convolution with an identity filter)
- In one case, it clearly isn't invertible (e.g. convolution with an all zero filter)
- What about for common filters like a Gaussian?

But you can't invert multiplication by 0

- But it's not quite zero, is it...



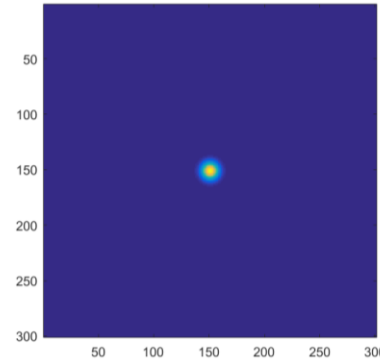
Let's experiment on Novak



Convolution



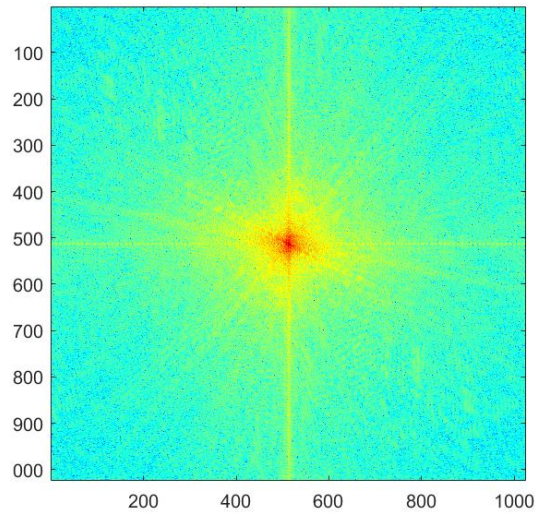
*



=

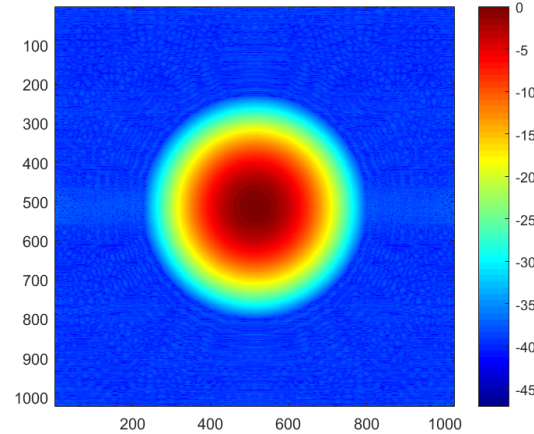


FFT ↓



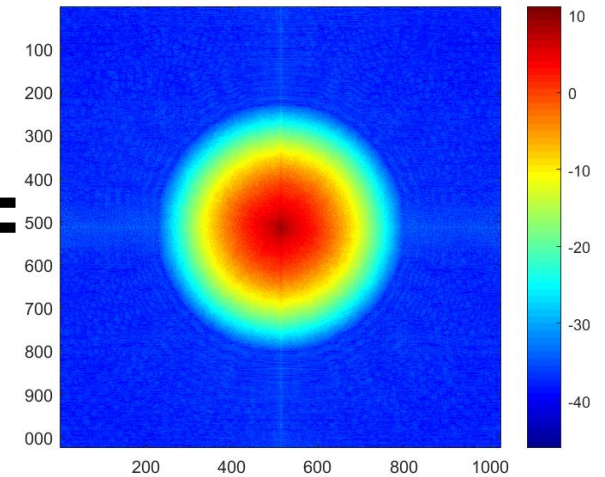
*

FFT ↓

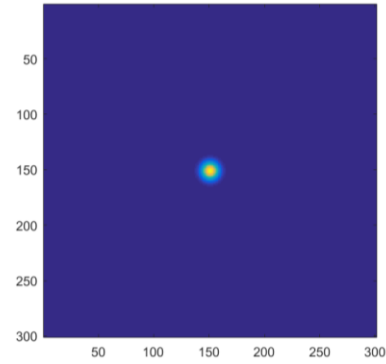


=

iFFT ↑



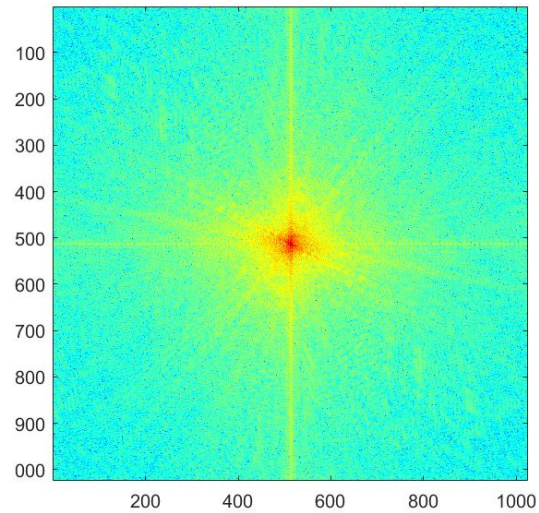
Deconvolution?



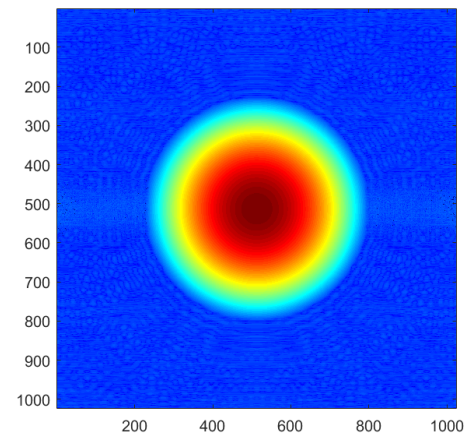
iFFT 

FFT 

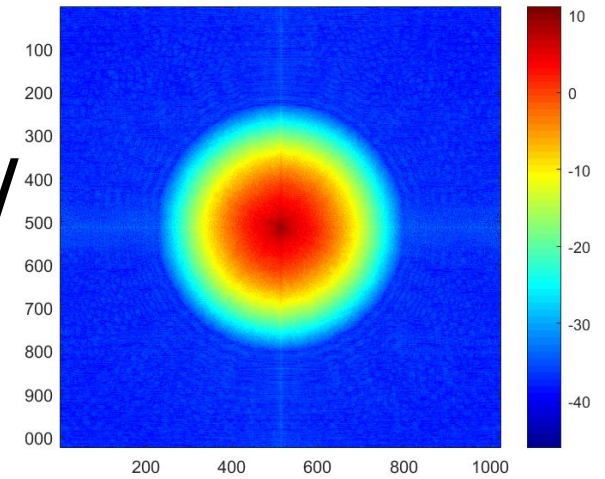
FFT 



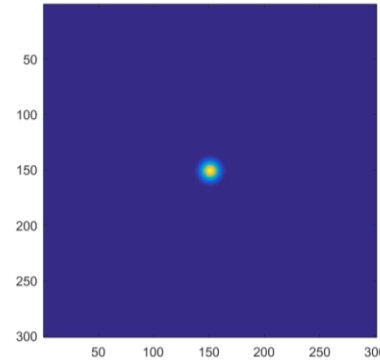
=



/



But under more realistic conditions



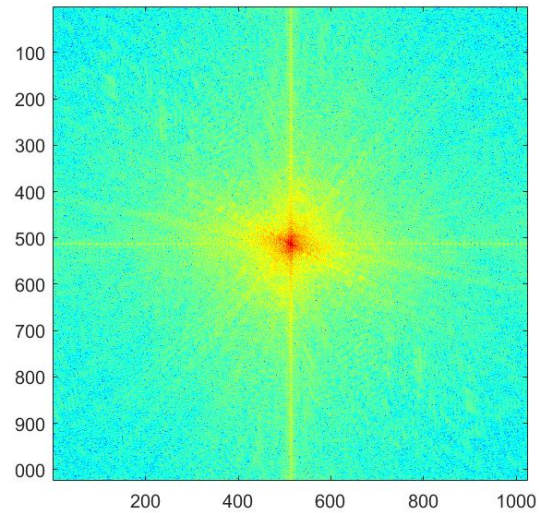
Random noise, .000001 magnitude



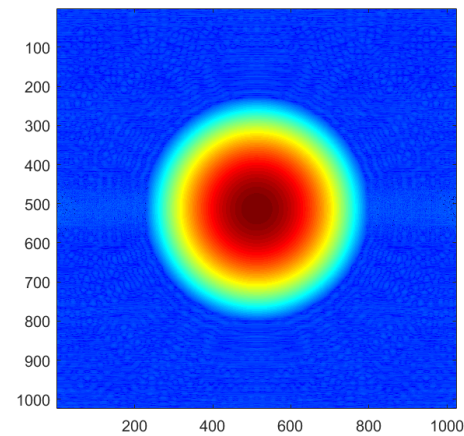
iFFT ↑

FFT ↓

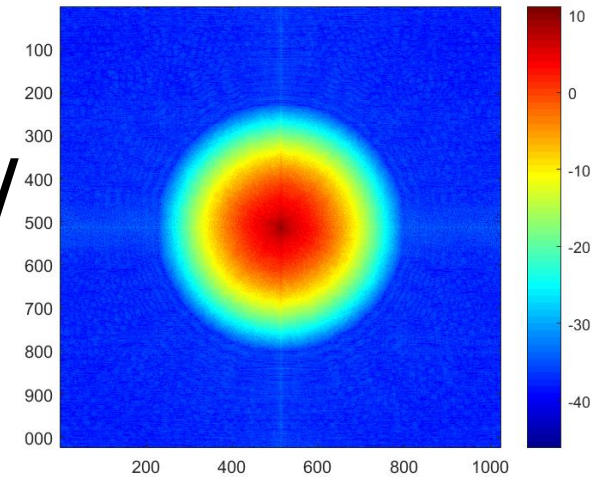
FFT ↓



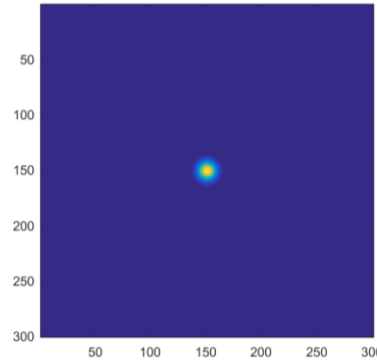
=



/



But under more realistic conditions



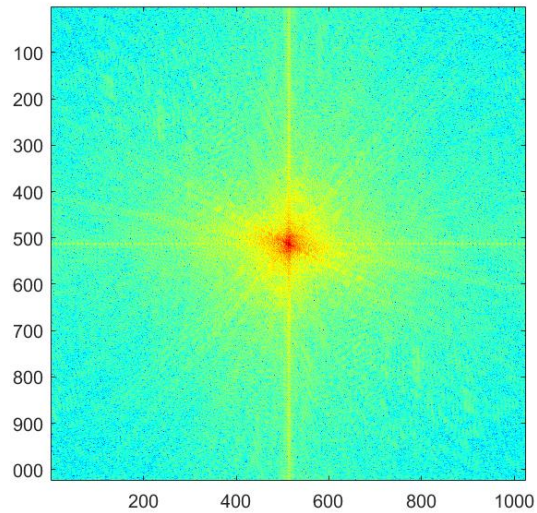
Random noise, .0001 magnitude



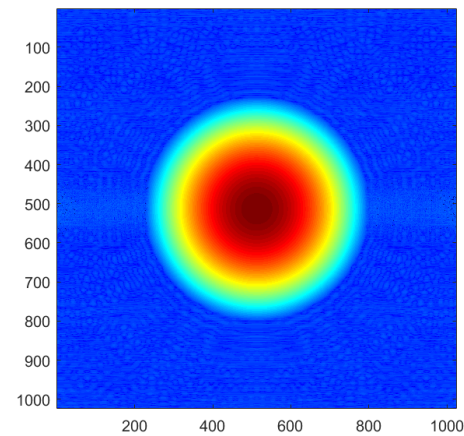
iFFT ↑

FFT ↓

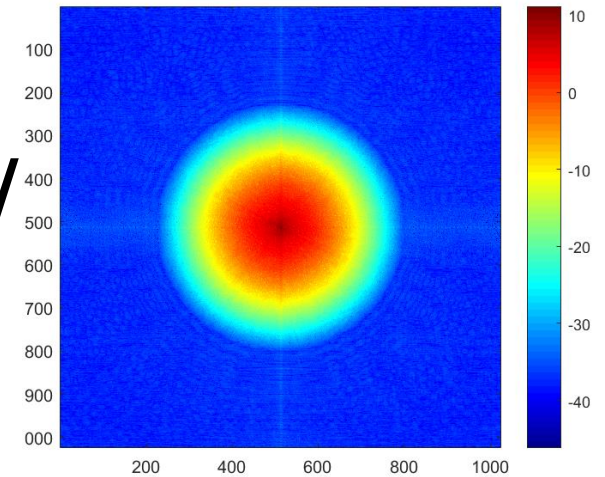
FFT ↓



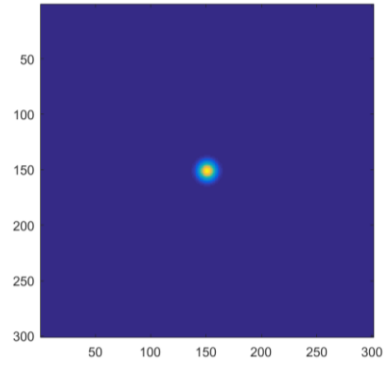
=



/



But under more realistic conditions



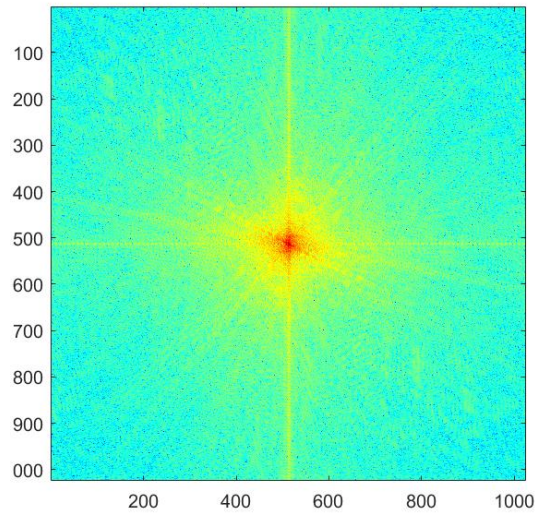
Random noise, .001 magnitude



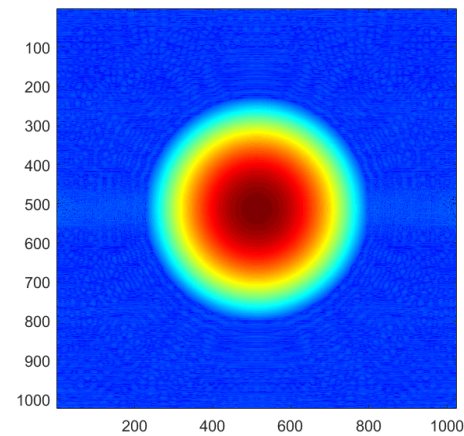
iFFT ↑

FFT ↓

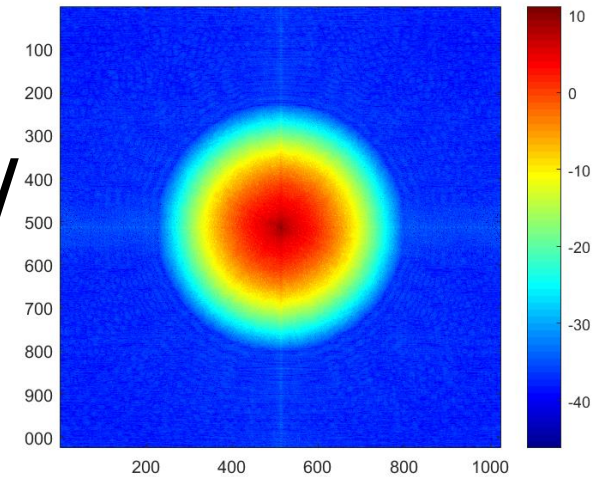
FFT ↓



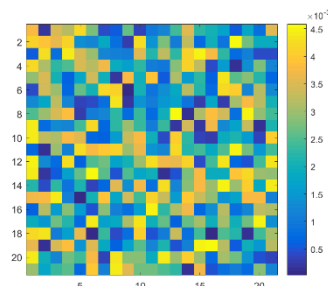
=



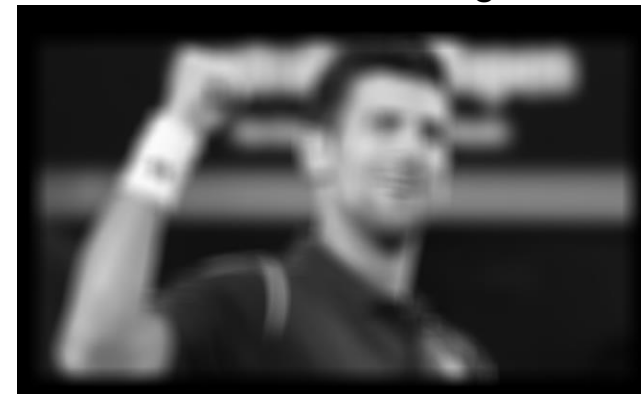
/



With a random filter...



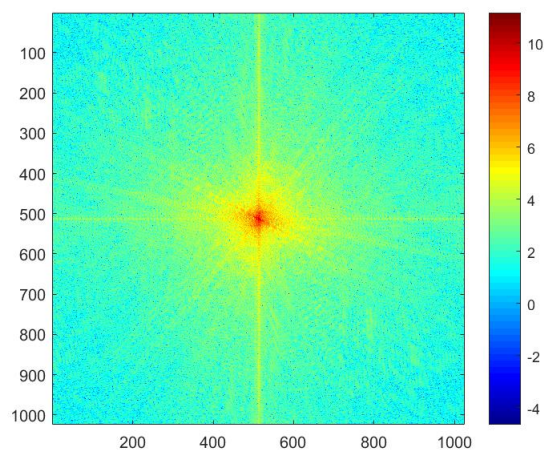
Random noise, .001 magnitude



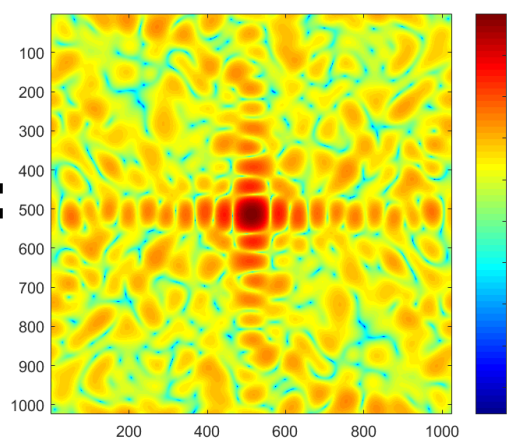
iFFT 

FFT 

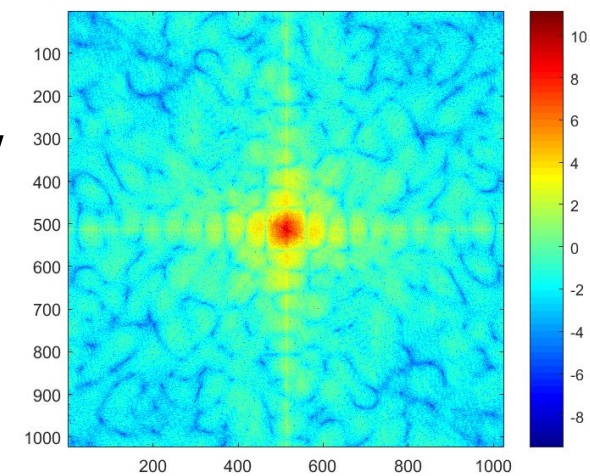
FFT 



=



/



Deconvolution is hard

- Active research area.
- Even if you know the filter (non-blind deconvolution), it is still very hard and requires strong *regularization*.
- If you don't know the filter (blind deconvolution) it is harder still.

Blind Deconvolution Example

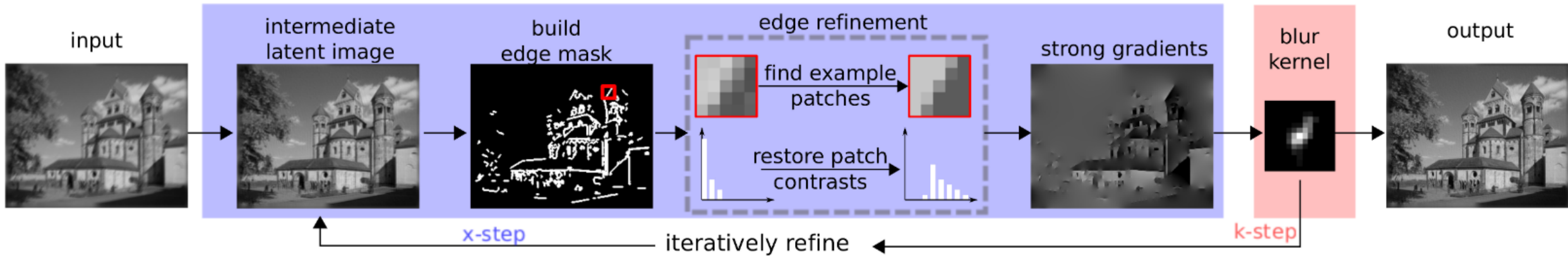
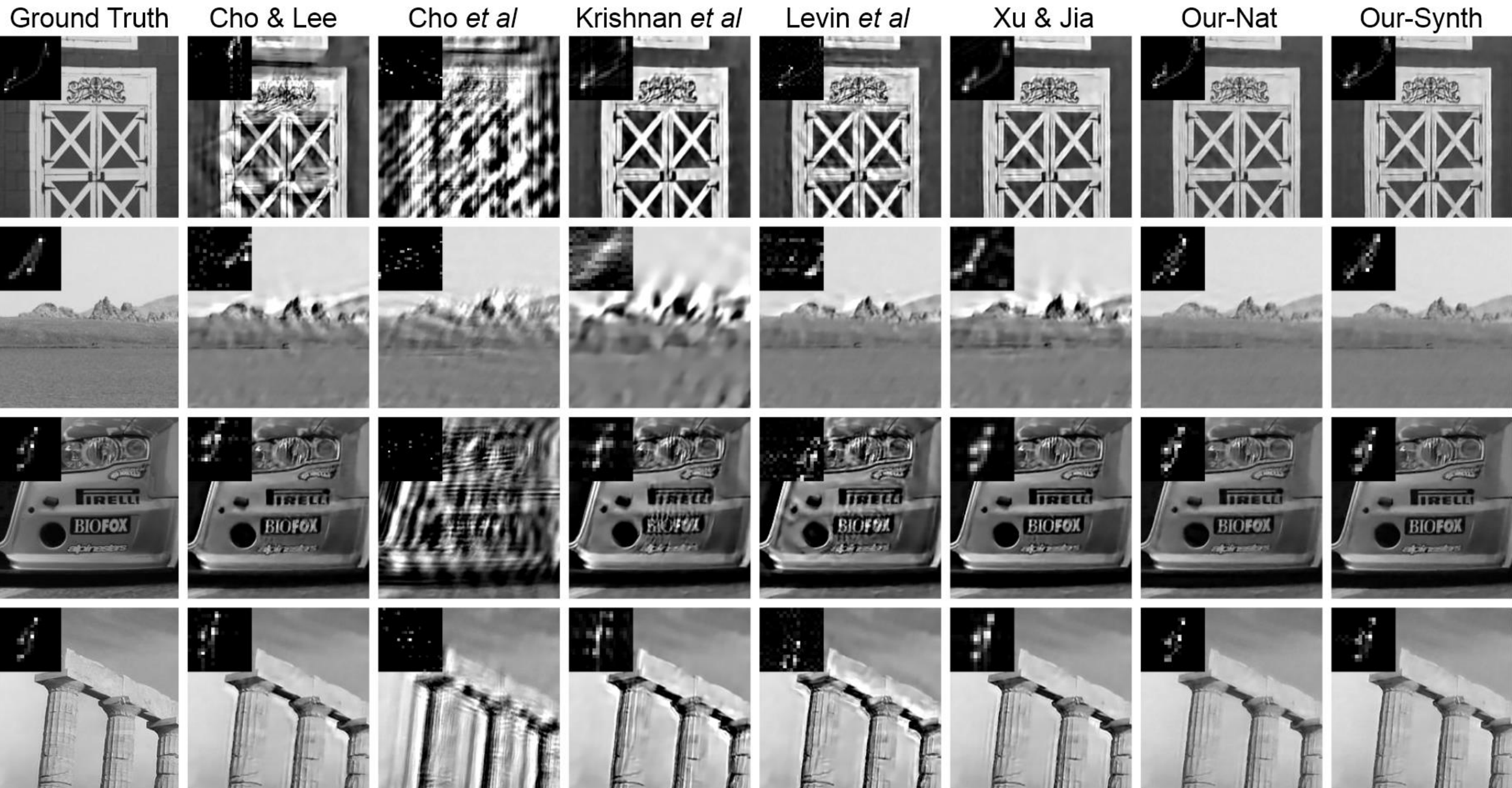


Figure 1. Algorithm pipeline. Our algorithm iterates between x -step and k -step with the help of a patch prior for edge refinement process. In particular, we coerce edges to become sharp and increase local contrast for edge patches. The blur kernel is then updated using the strong gradients from the restored latent image. After kernel estimation, the method of [20] is used for final non-blind deconvolution.



Edge-based Blur Kernel Estimation Using Patch Priors.
 Libin Sun, Sunghyun Cho, Jue Wang, and James Hays.
 IEEE International Conference on Computational Photography 2013.