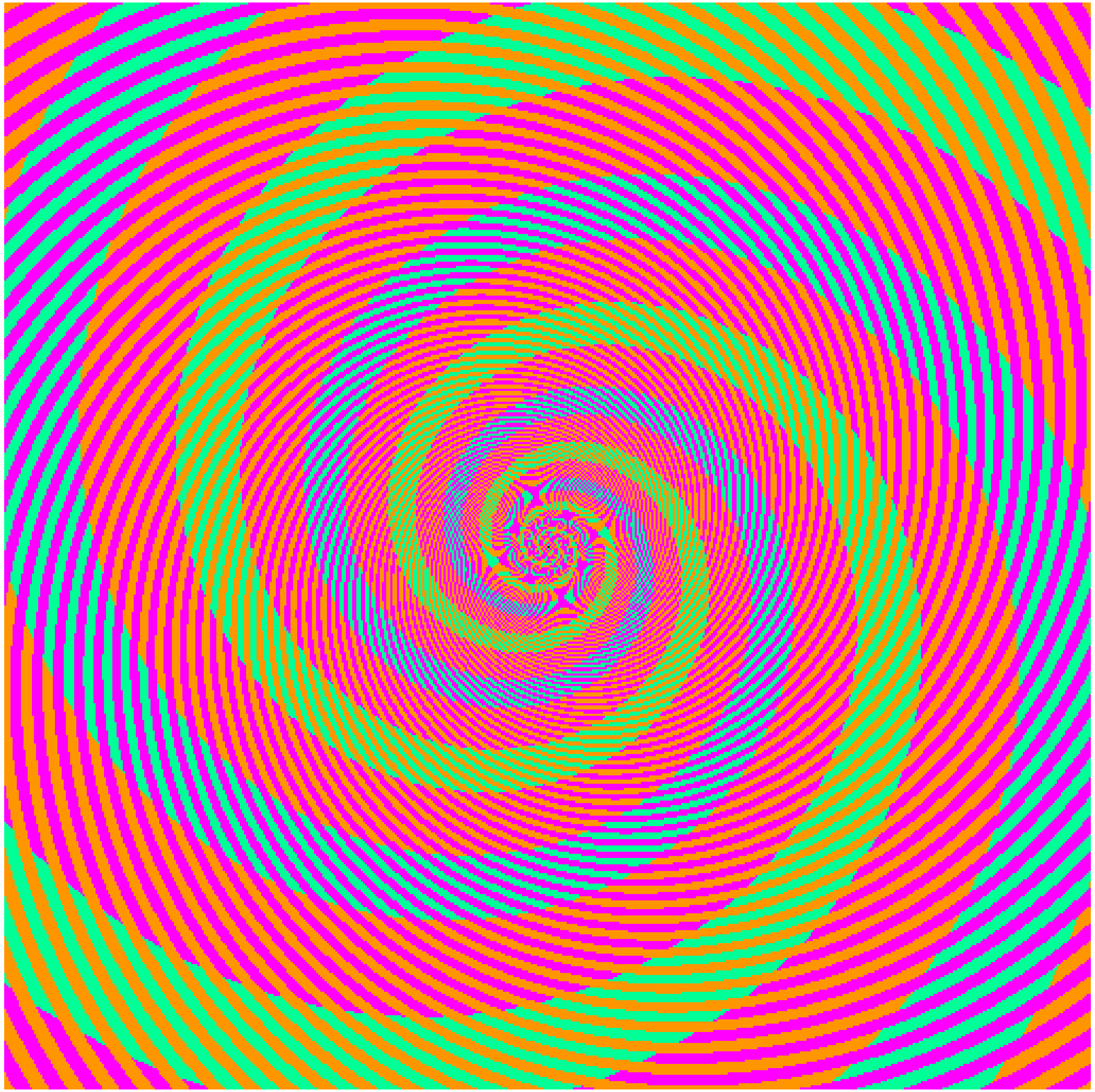
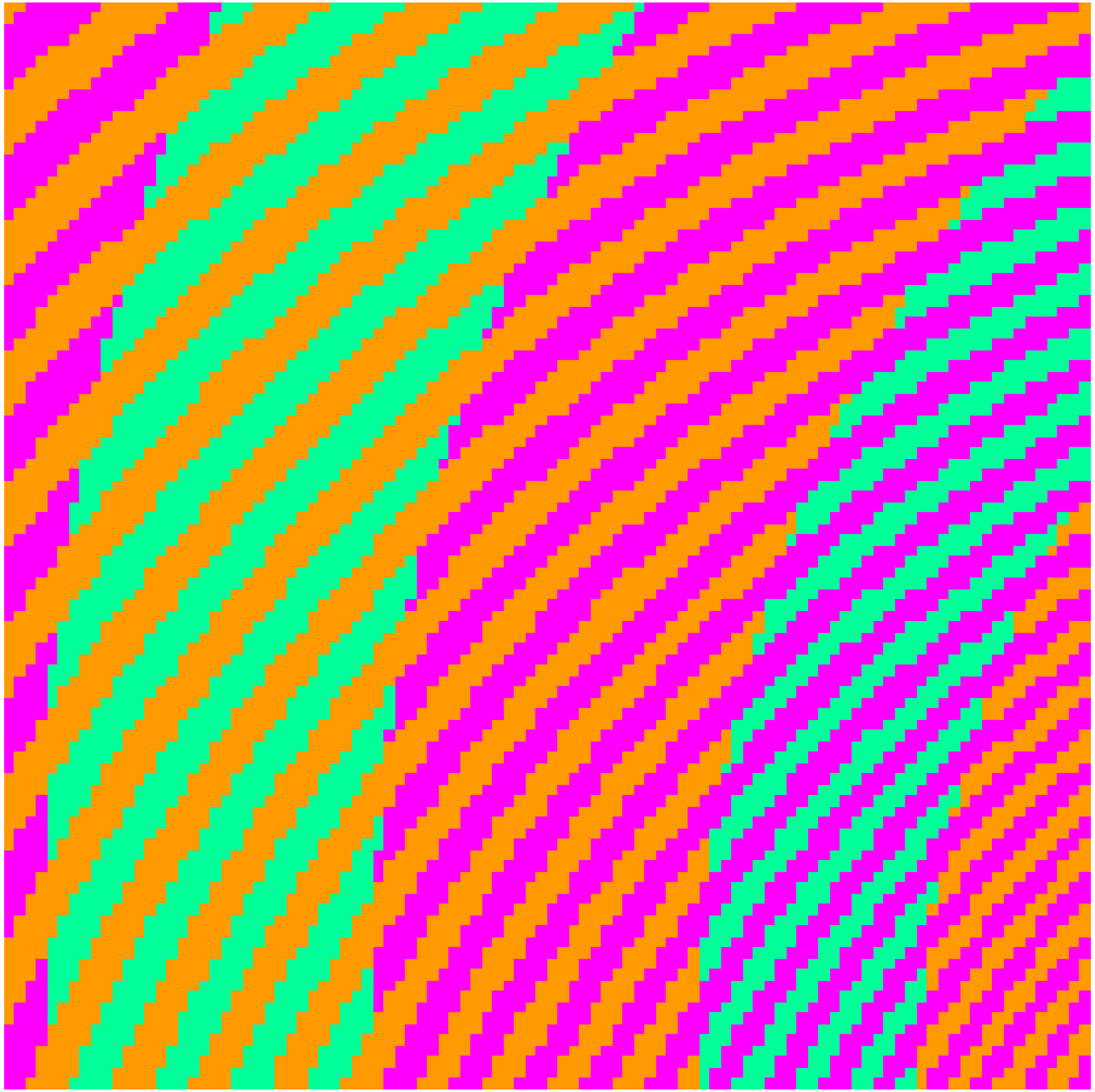
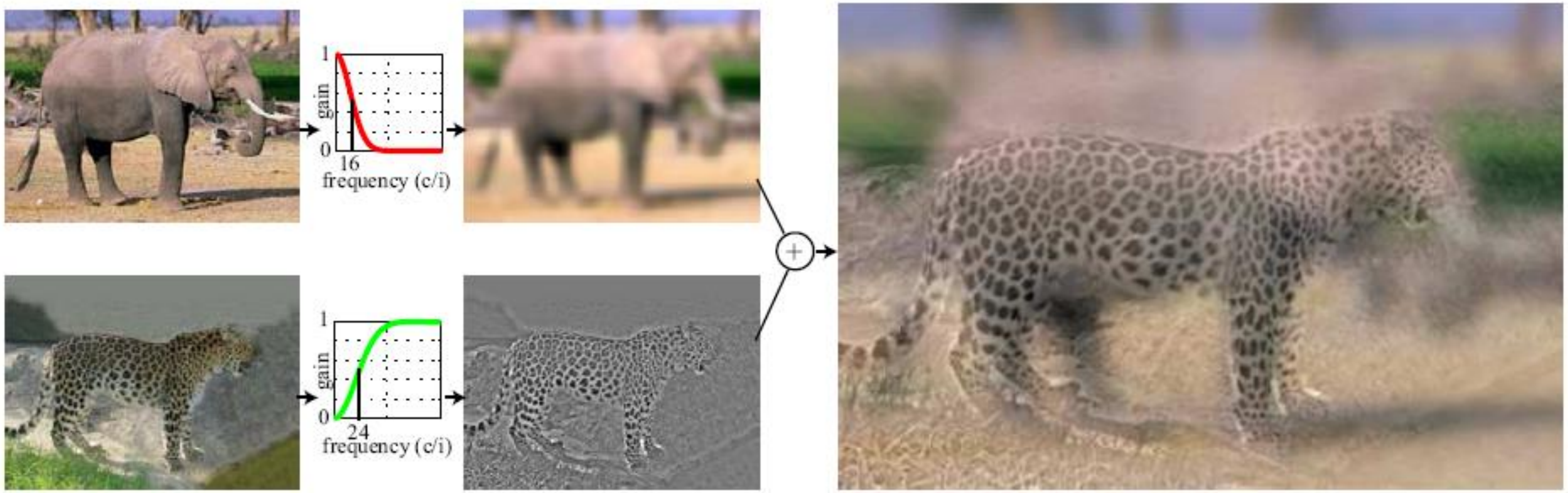


The blue and green colors are actually the same



Hybrid Images



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

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

?

Slide: Hoiem

Recap of Filtering Discussion from Previous Lecture

- Linear filtering is dot product at each position
  - Not a matrix multiplication
  - Can smooth, sharpen, translate (among many other uses)
  - Linear filters "*look for" features that resemble the filter itself*

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

- Be aware of details for filter size, extrapolation, cropping

Why do we care so much about filtering/convolution?

- Pixels are individually weak and noisy signals. Reasoning over neighborhoods helps.

Surely there are other ways to extract information from images?

- Yes, but they may be more brittle, slower to compute, or less easy to plug into machine learning tools.

Alternative to Filtering - Viola Jones Face Detection

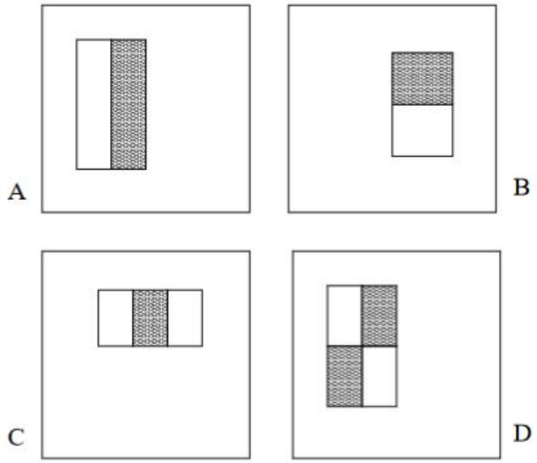


Figure 1: Example rectangle features shown relative to the enclosing detection window. The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the grey rectangles. Two-rectangle features are shown in (A) and (B). Figure (C) shows a three-rectangle feature, and (D) a four-rectangle feature.

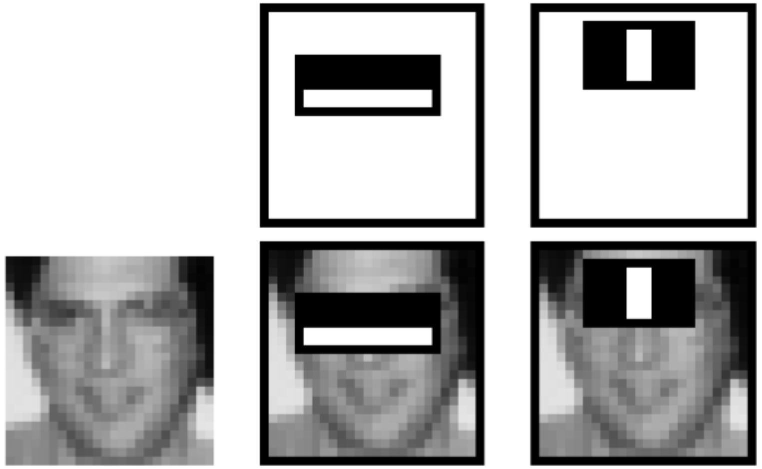
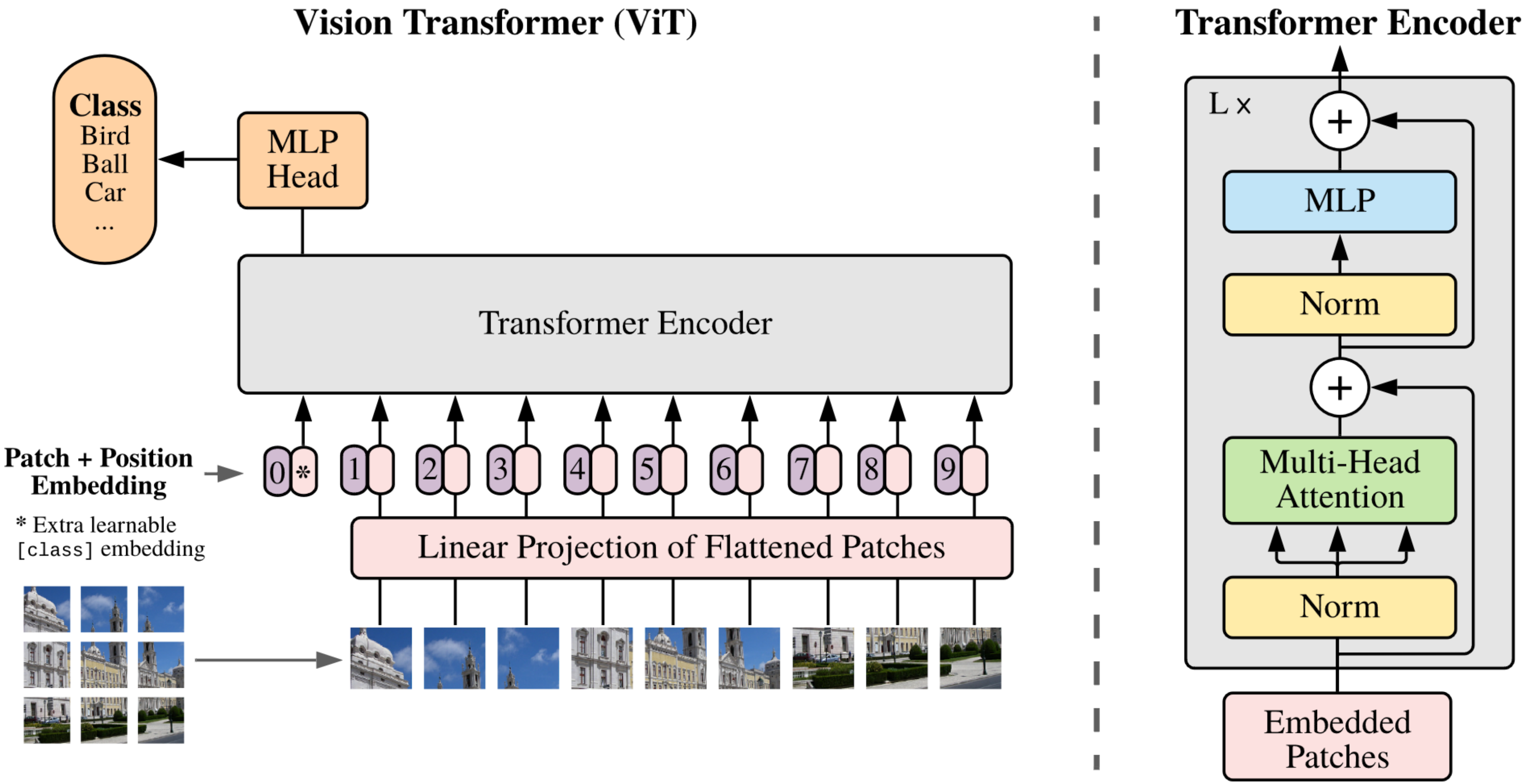


Figure 3: The first and second features selected by AdaBoost. The two features are shown in the top row and then overlayed on a typical training face in the bottom row. The first feature measures the difference in intensity between the region of the eyes and a region across the upper cheeks. The feature capitalizes on the observation that the eye region is often darker than the cheeks. The second feature compares the intensities in the eye regions to the intensity across the bridge of the nose.

Viola, Jones. Rapid object detection using a boosted cascade of simple features. CVPR 2001.

Alternative to Filtering – Non-overlapping Patches



An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale
A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. International Conference on Learning Representations, (2021)

Filtering vs. Convolution

```
f=filter, size k x l
I=image,  size m x n
h=output, size m x n
```

- 2d filtering

$$h[m,n] = \sum_{k,l} f[k,l]\, I[m+k, n+l]$$

- 2d convolution

$$h[m,n] = \sum_{k,l} f[k,l]\, I[m-k, n-l]$$

In Python you can use https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.convolve2d.html

Key properties of linear filters

Linearity:

```
imfilter(I, f1 + f2) =
   imfilter(I,f1) + imfilter(I,f2)
```

Shift invariance: same behavior regardless of pixel location

```
imfilter(I,shift(f)) = shift(imfilter(I,f))
```

Any linear, shift-invariant operator can be represented as a convolution

Source: S. Lazebnik

More properties

- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal
  - But particular filtering implementations might break this equality
- Associative: $a * (b * c) = (a * b) * c$
  - Often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - This is equivalent to applying one filter: a $* (b_1 * b_2 * b_3)$
- Distributes over addition: $a * (b + c) = (a * b) + (a * c)$
- Scalars factor out: $ka * b = a * kb = k\,(a * b)$
- Identity: unit impulse $e = [0, 0, 1, 0, 0]$,
  $a * e = a$

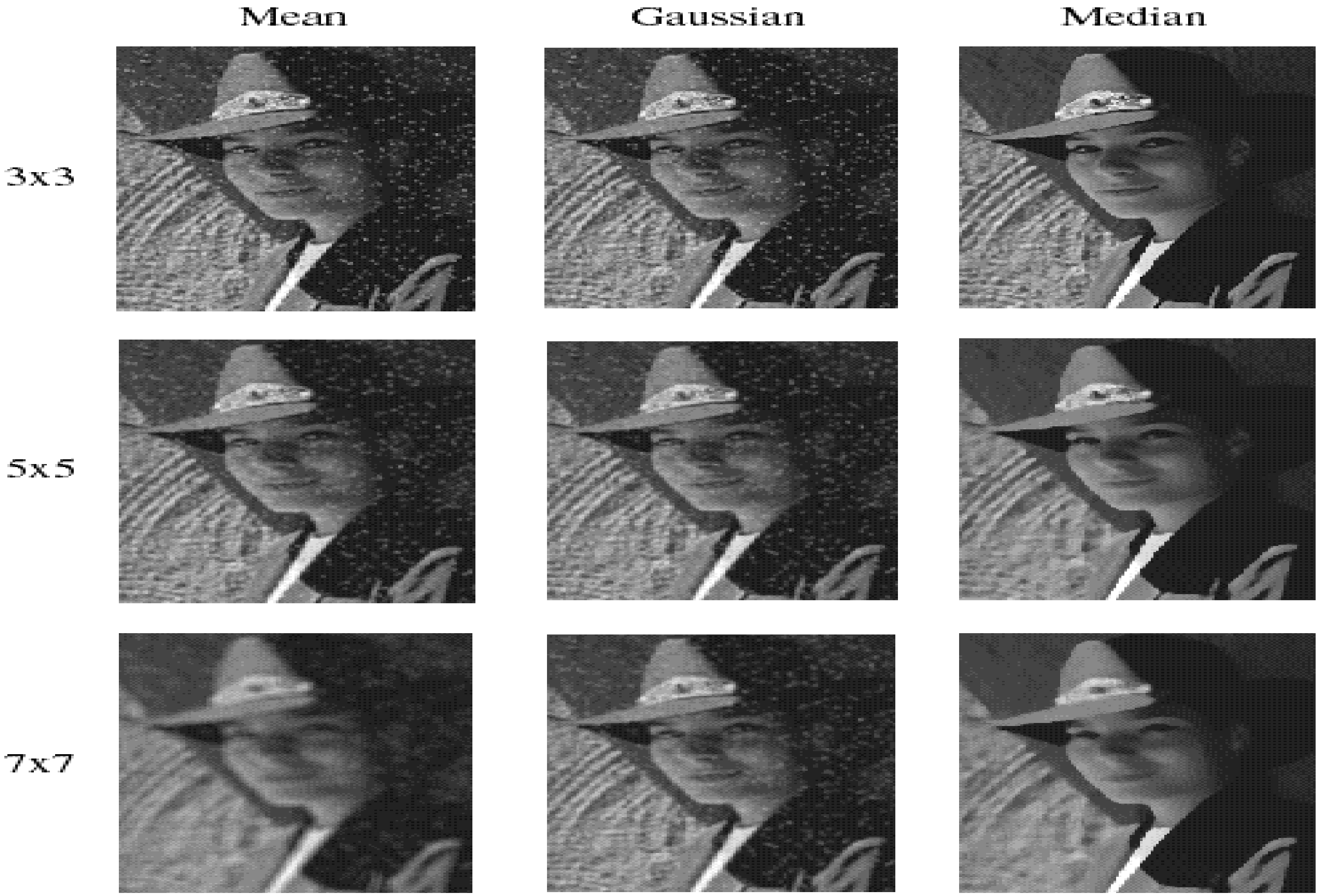
Source: S. Lazebnik

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?

Slide by Steve Seitz

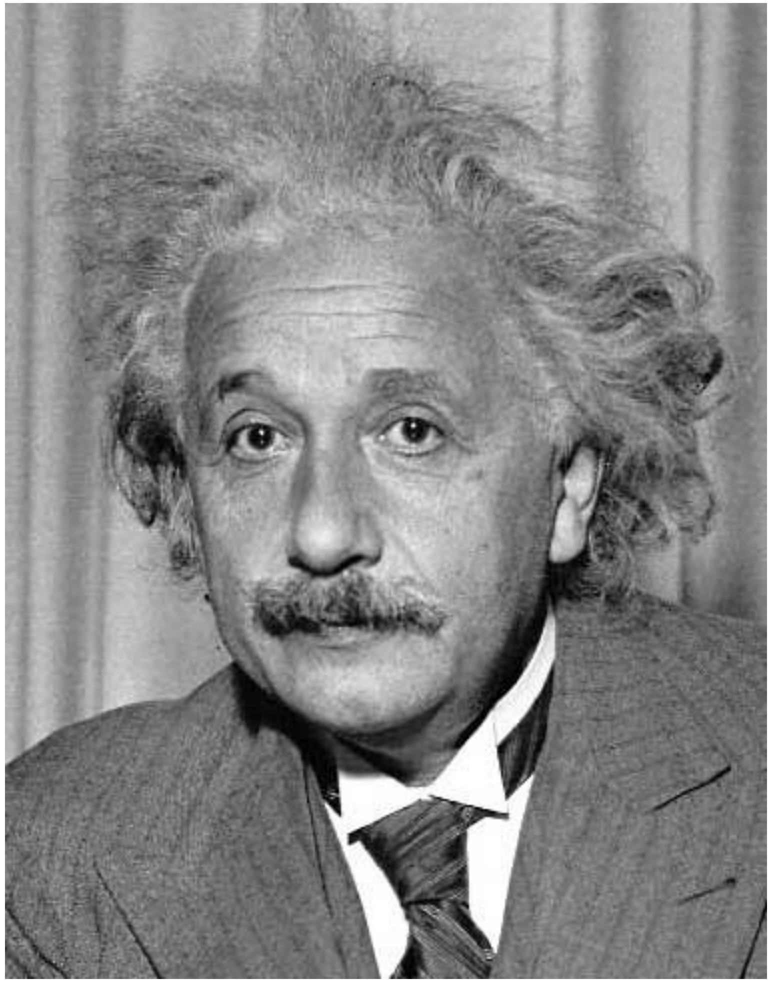
Comparison: salt and pepper noise



Slide by Steve Seitz

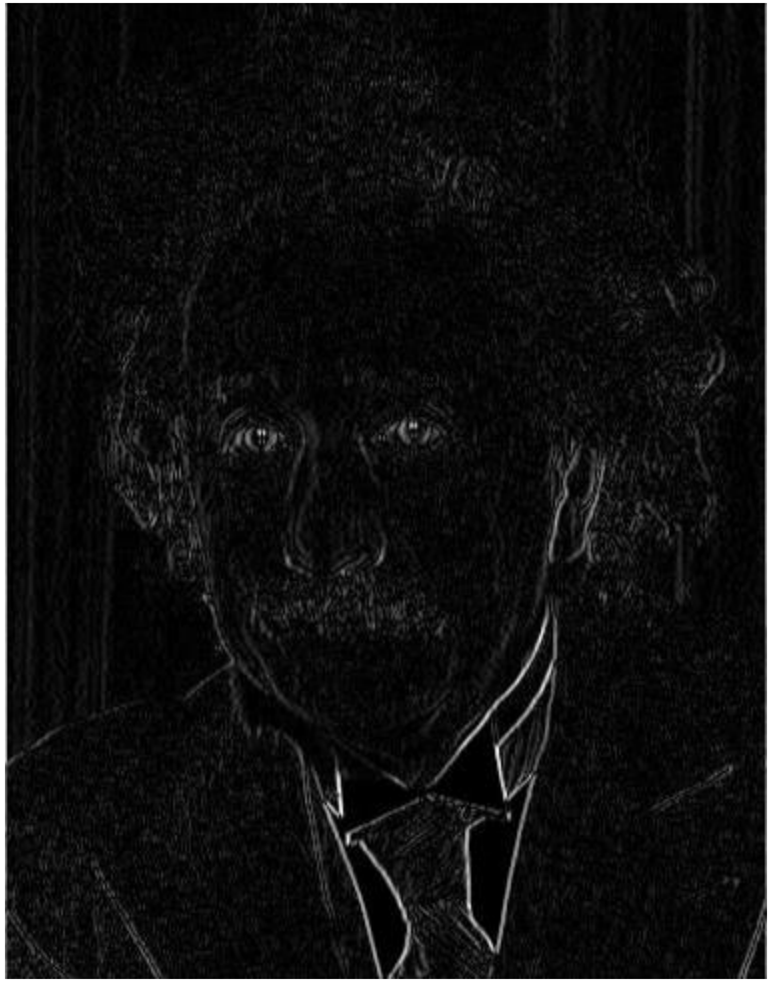
Can we do edge detection in a single filtering step?

Sobel filters



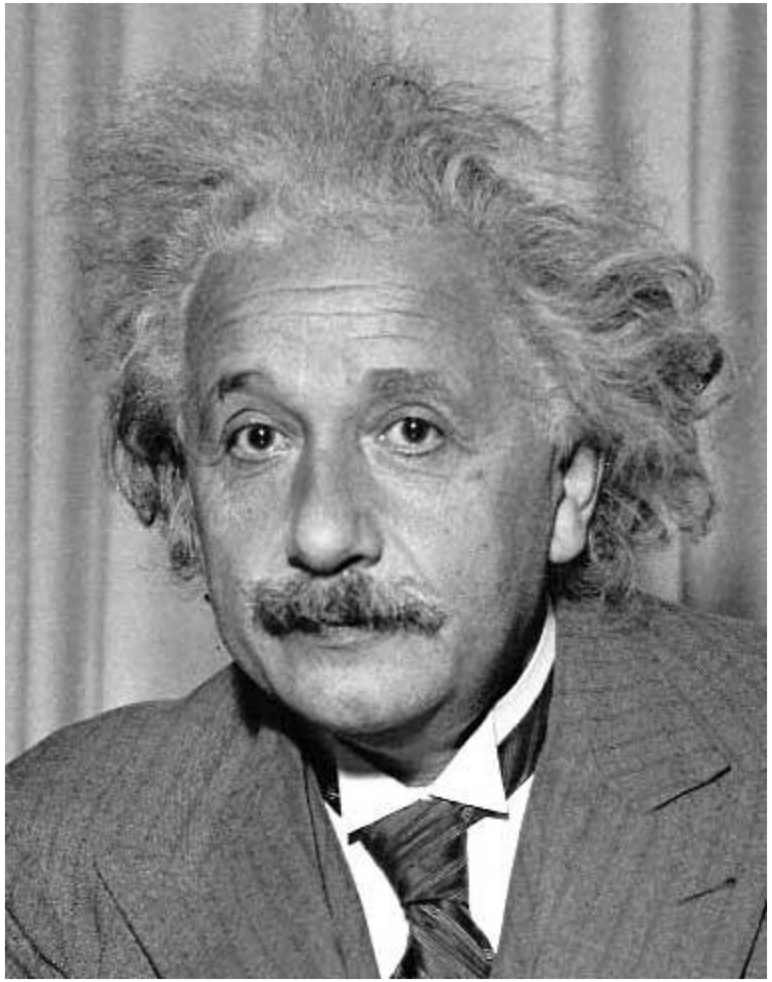
1	0	-1
2	0	-2
1	0	-1

Sobel



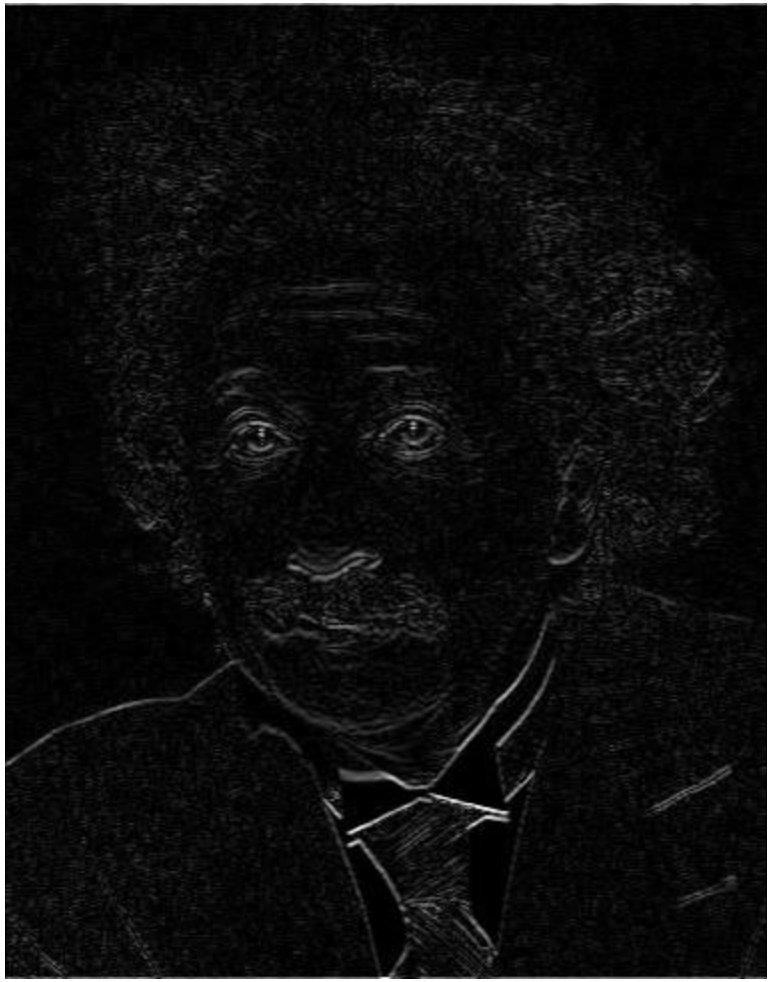
Vertical Edge
(absolute value)

Sobel filters



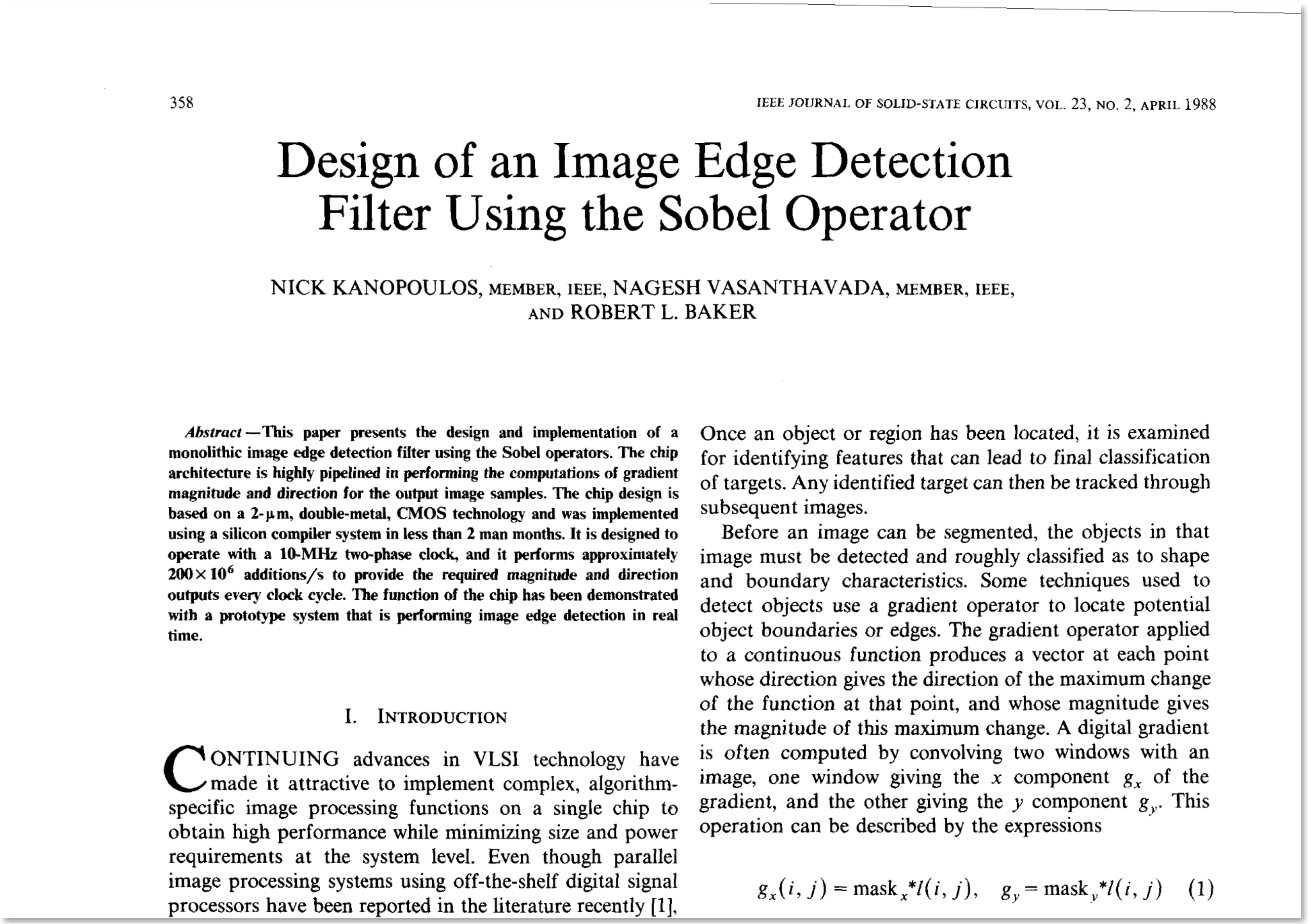
1	2	1
0	0	0
-1	-2	-1

Sobel



Horizontal Edge
(absolute value)

Sobel Filters for Edge Detection



358 IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 23, NO. 2, APRIL 1988

Design of an Image Edge Detection Filter Using the Sobel Operator

NICK KANOPOULOS, MEMBER, IEEE, NAGESH VASANTHAVADA, MEMBER, IEEE, AND ROBERT L. BAKER

Abstract—This paper presents the design and implementation of a monolithic image edge detection filter using the Sobel operators. The chip architecture is highly pipelined in performing the computations of gradient magnitude and direction for the output image samples. The chip design is based on a 2-μm, double-metal, CMOS technology and was implemented using a silicon compiler system in less than 2 man months. It is designed to operate with a 10-MHz two-phase clock, and it performs approximately 200×10^6 additions/s to provide the required magnitude and direction outputs every clock cycle. The function of the chip has been demonstrated with a prototype system that is performing image edge detection in real time.

I. Introduction

CONTINUING advances in VLSI technology have made it attractive to implement complex, algorithm-specific image processing functions on a single chip to obtain high performance while minimizing size and power requirements at the system level. Even though parallel image processing systems using off-the-shelf digital signal processors have been reported in the literature recently [1],

Once an object or region has been located, it is examined for identifying features that can lead to final classification of targets. Any identified target can then be tracked through subsequent images.

Before an image can be segmented, the objects in that image must be detected and roughly classified as to shape and boundary characteristics. Some techniques used to detect objects use a gradient operator to locate potential object boundaries or edges. The gradient operator applied to a continuous function produces a vector at each point whose direction gives the direction of the maximum change of the function at that point, and whose magnitude gives the magnitude of this maximum change. A digital gradient is often computed by convolving two windows with an image, one window giving the x component g_x of the gradient, and the other giving the y component g_y. This operation can be described by the expressions

$$g_x(i,j) = \text{mask}_x * I(i,j), \quad g_y = \text{mask}_y * I(i,j) \qquad (1)$$

https://ieeexplore.ieee.org/document/996 the magnitude of this maximum change. A digital gradient is often computed by convolving two windows with an image, one window giving the x component g_x of the gradient, and the other giving the y component g_y. This operation can be described by the expressions

$$g_x(i, j) = \text{mask}_x * l(i, j), \quad g_y = \text{mask}_y * l(i, j) \qquad (1)$$

where $l(i, j)$ is indicating some neighborhood of pixel (i, j), and * denotes convolution. Fig. 2 illustrates an example of this operation.

$$E_H = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * I$$

$$E_V = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * I$$

plifies the directional computation. The gradient magnitude

$$\text{Mag} = \left[E_H^2 + E_V^2\right]^{1/2} \qquad (2)$$

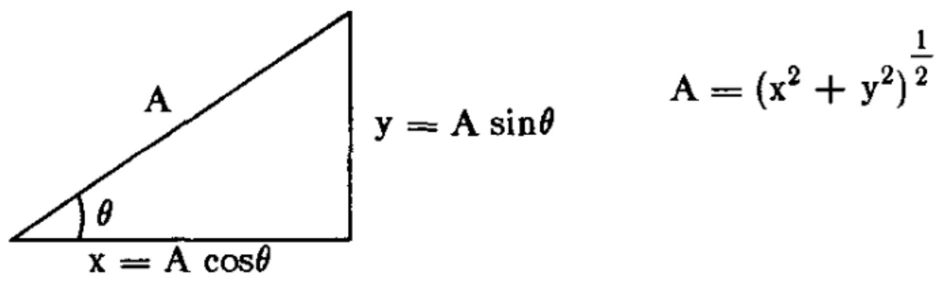


Fig. 3. Example for Euclidean norm approximation.

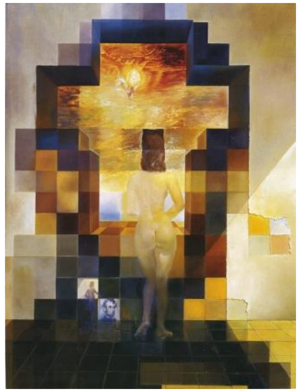
Review: questions

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 (and zero if they are the same)

2. Write down a filter that will compute the gradient in the x-direction:

```
gradx(y,x) = im(y,x+1)-im(y,x) for each x, y
```

Thinking in Frequency



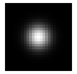
Slides: Hoiem, Efros, and others

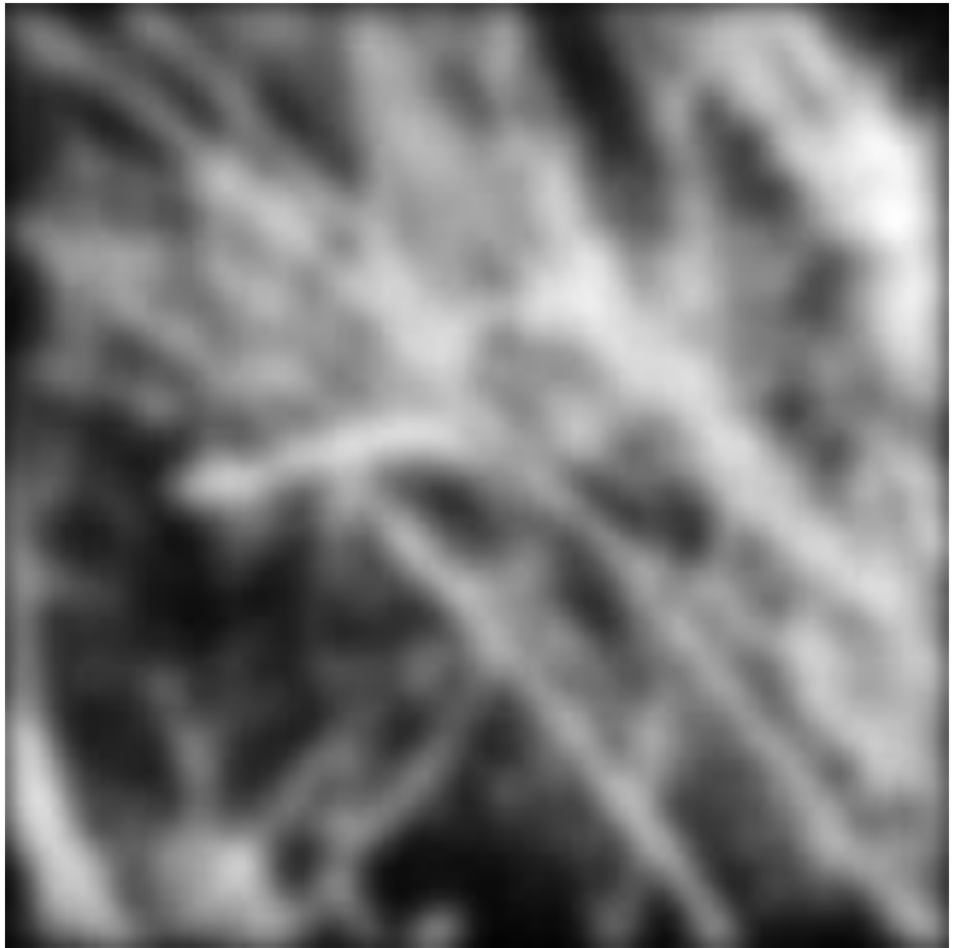
This lecture

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

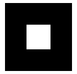
Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?

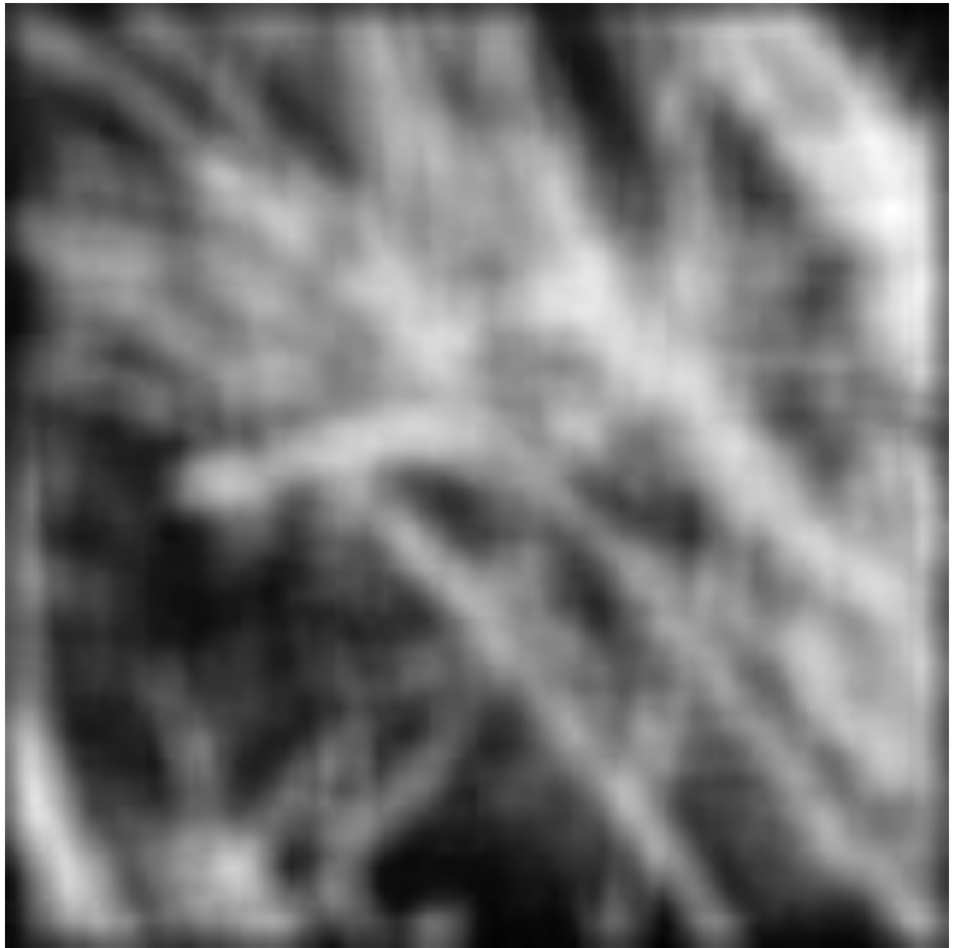
Gaussian





Box filter





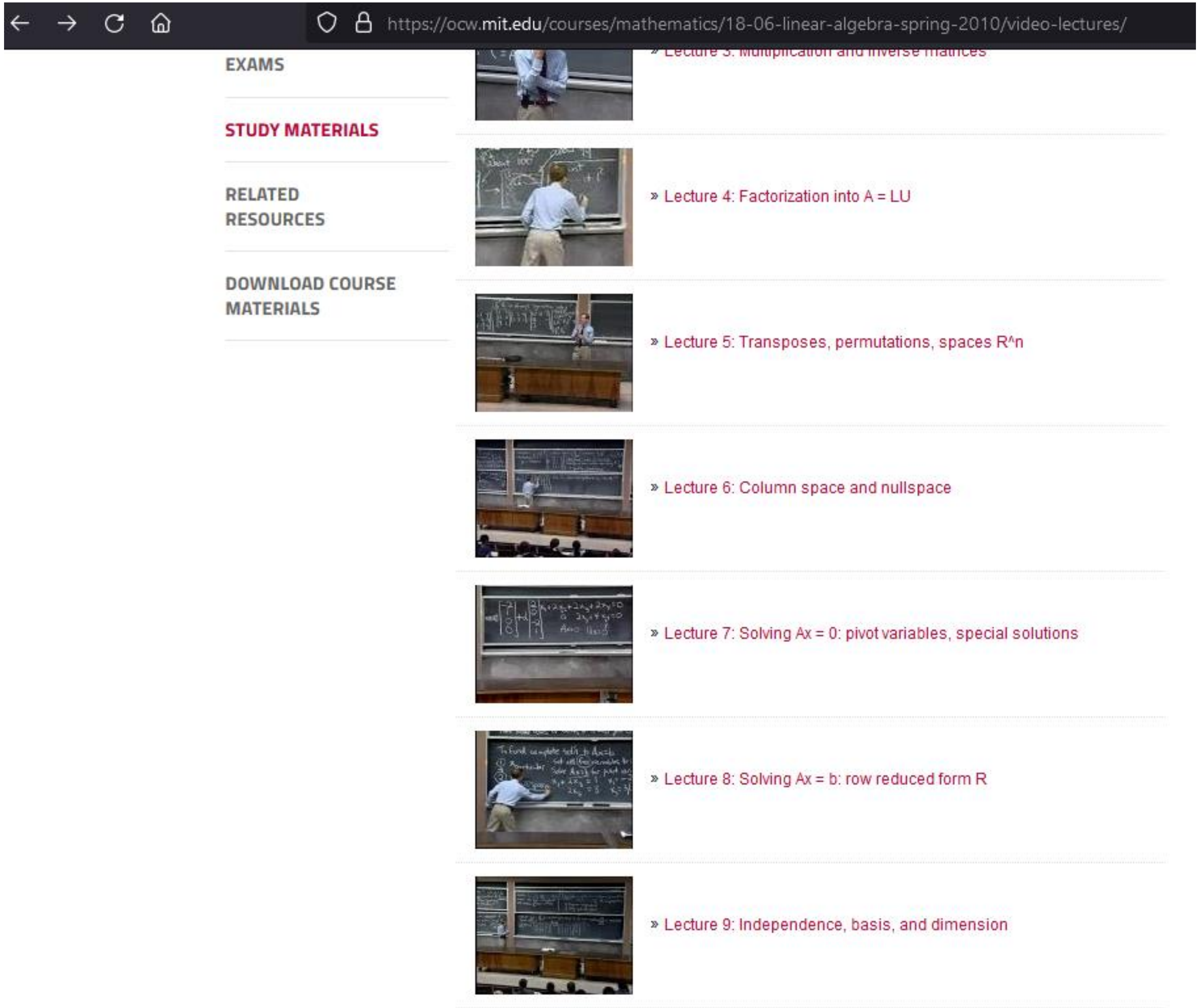
Why does a lower resolution image still make sense to us? What do we lose?

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

Slide: Hoiem

Thinking in terms of frequency

Background: Change of Basis



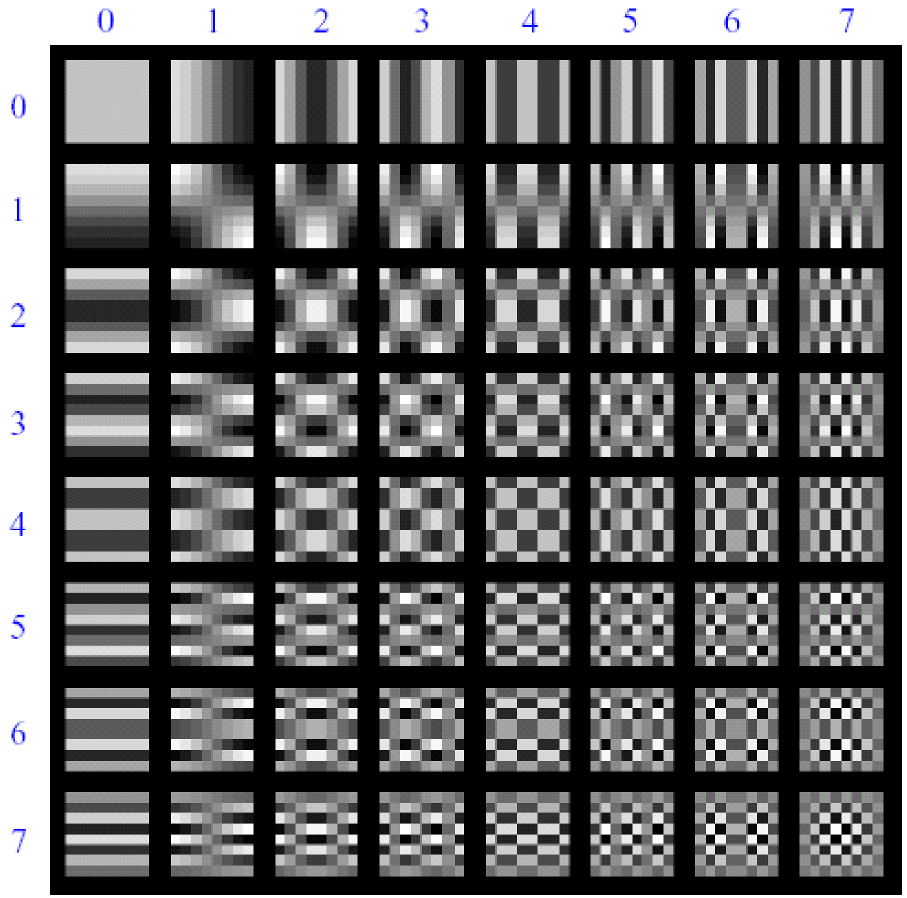
Background: Change of Basis

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)

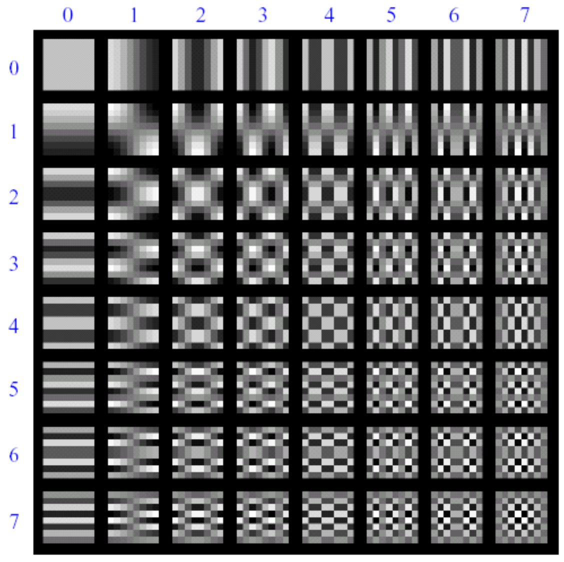


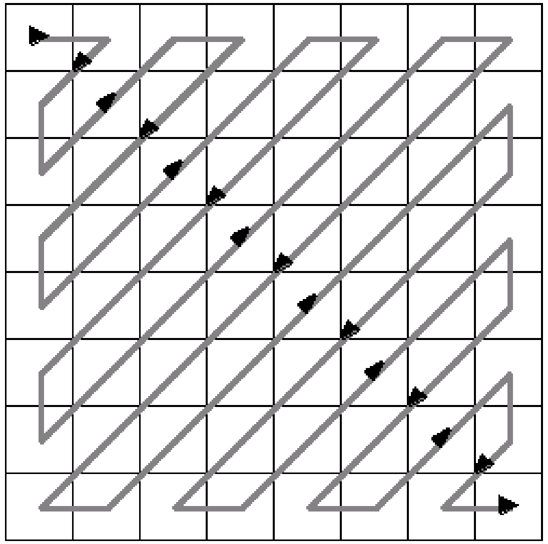
Block-based Discrete Cosine Transform (DCT)

https://en.wikipedia.org/wiki/JPEG

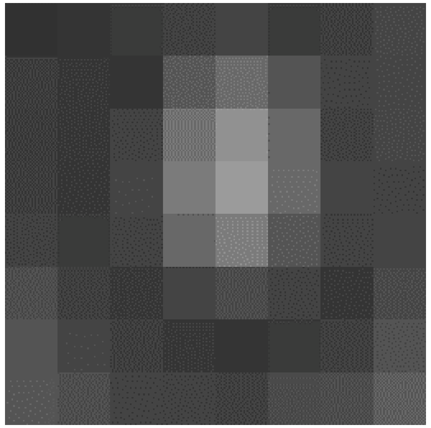
Using DCT in JPEG

- 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

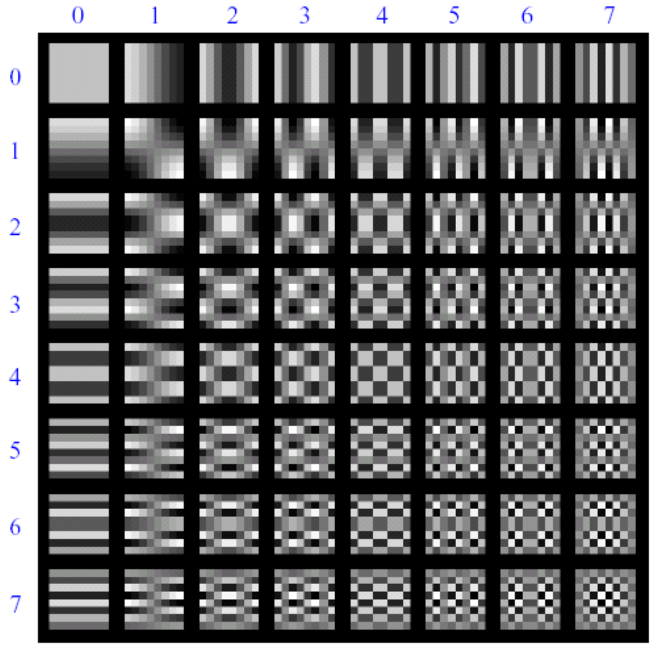




Lossy Image Compression (JPEG)



8x8 image patch



DCT bases

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

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

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

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 spatial sensitivity 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

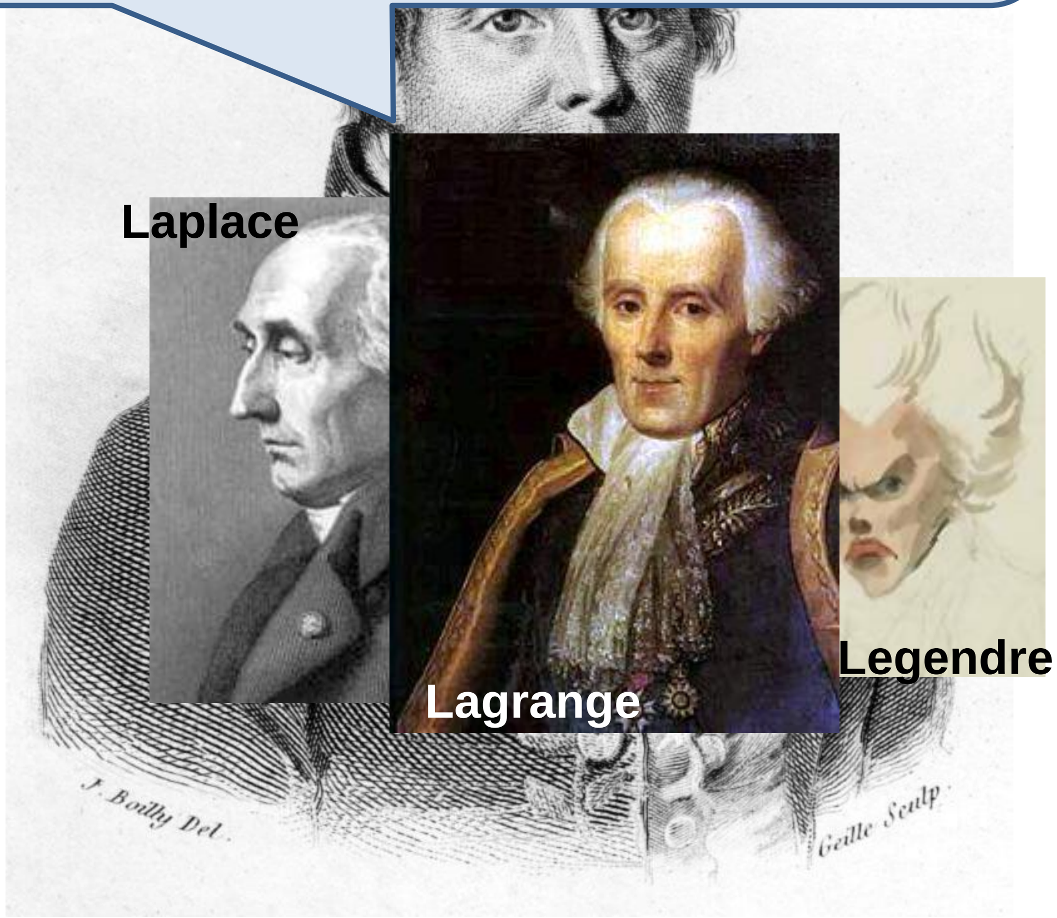
Jean Baptiste Joseph Fourier (1768-1830)

had crazy idea (1807):

> *...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.*

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

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



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

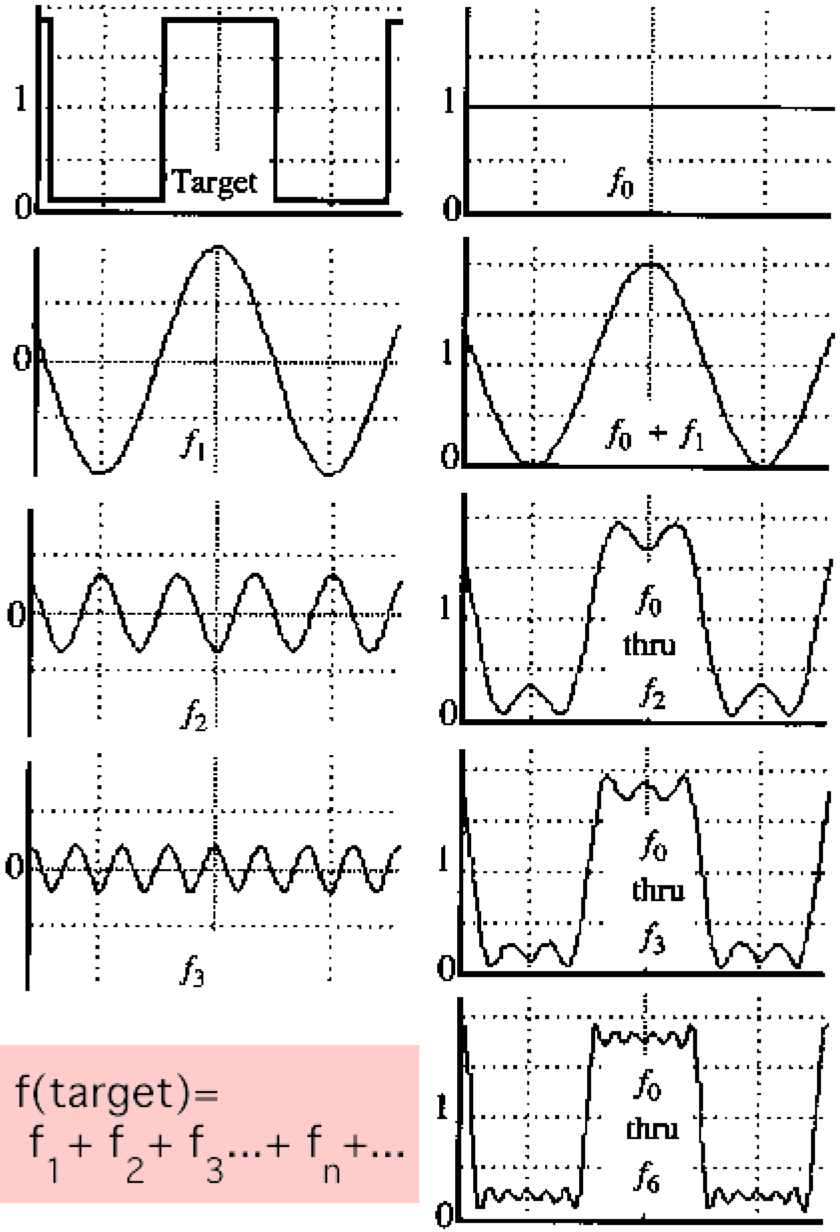
How would math have changed if the Slanket or Snuggie had been invented?

A sum of sines

Our building block:

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

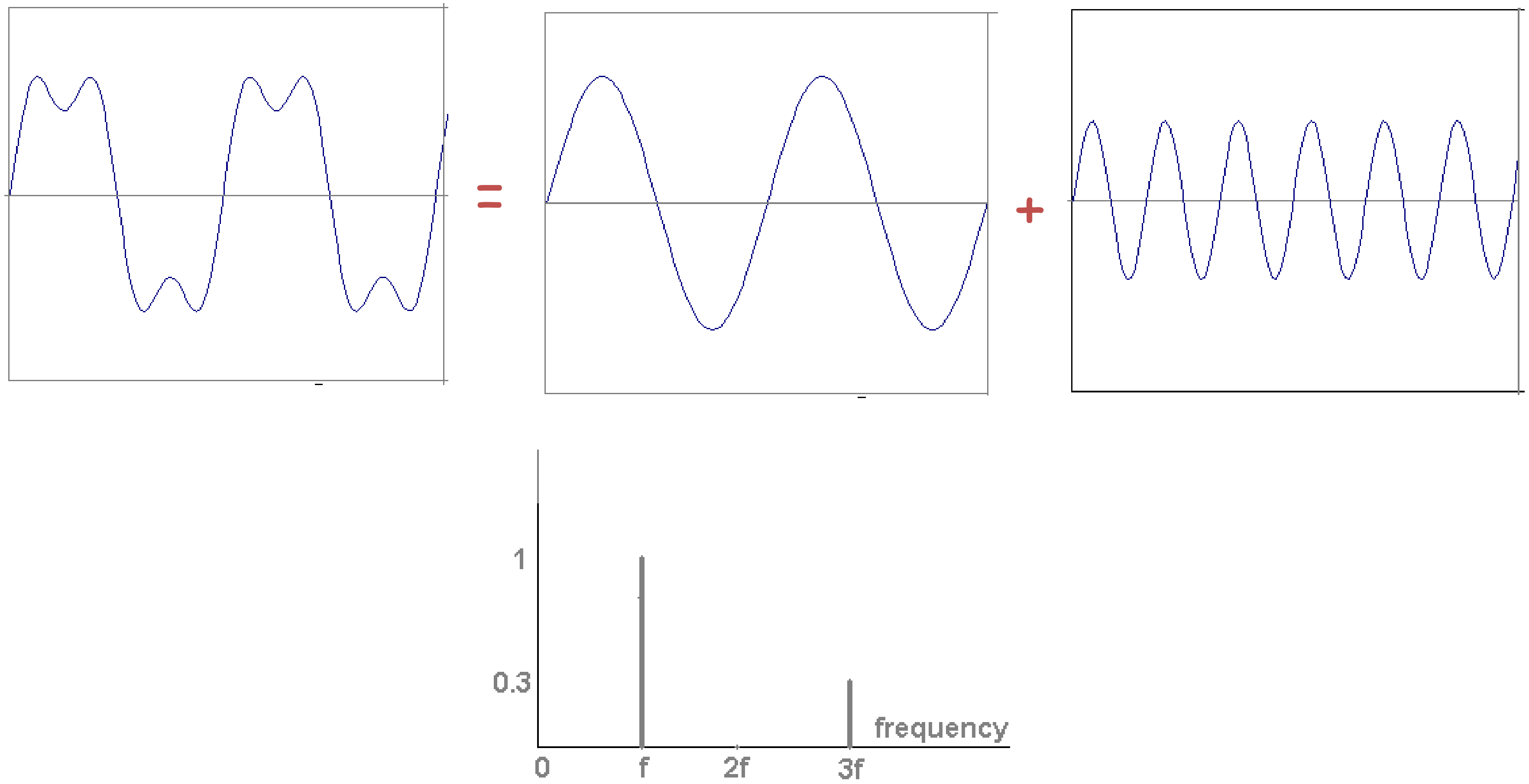
Add enough of them to get any signal *g(x)* you want!



f(target)=
$f_1 + f_2 + f_3 ... + f_n + ...$

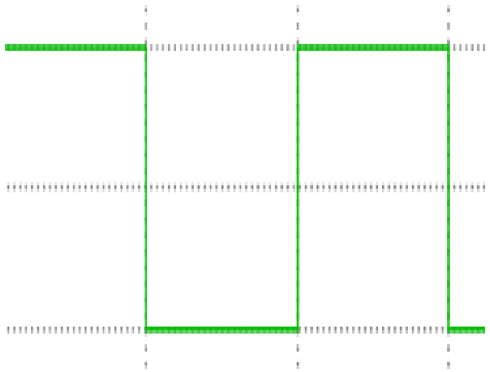
Frequency Spectra

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

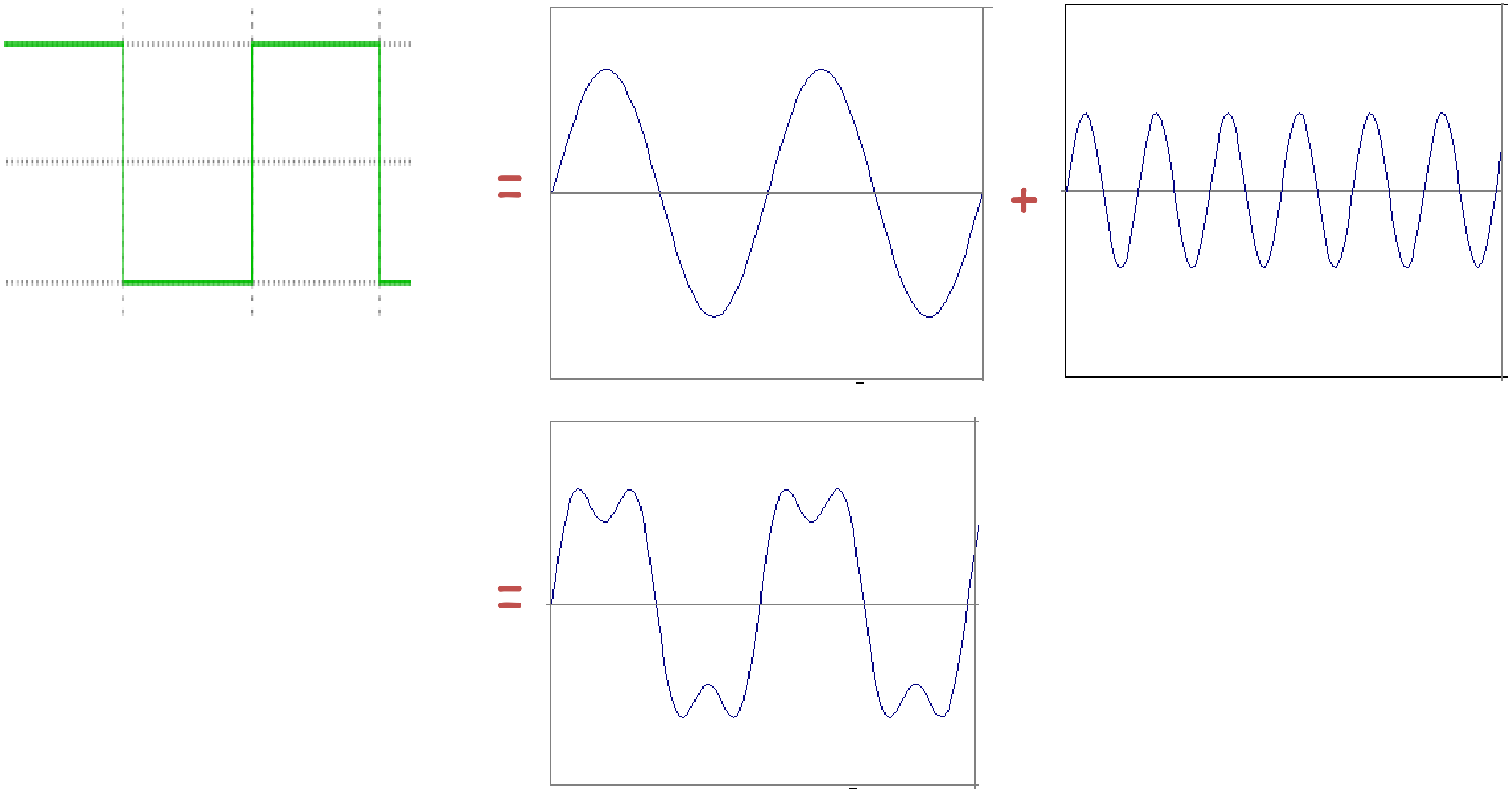


Slides: Efros

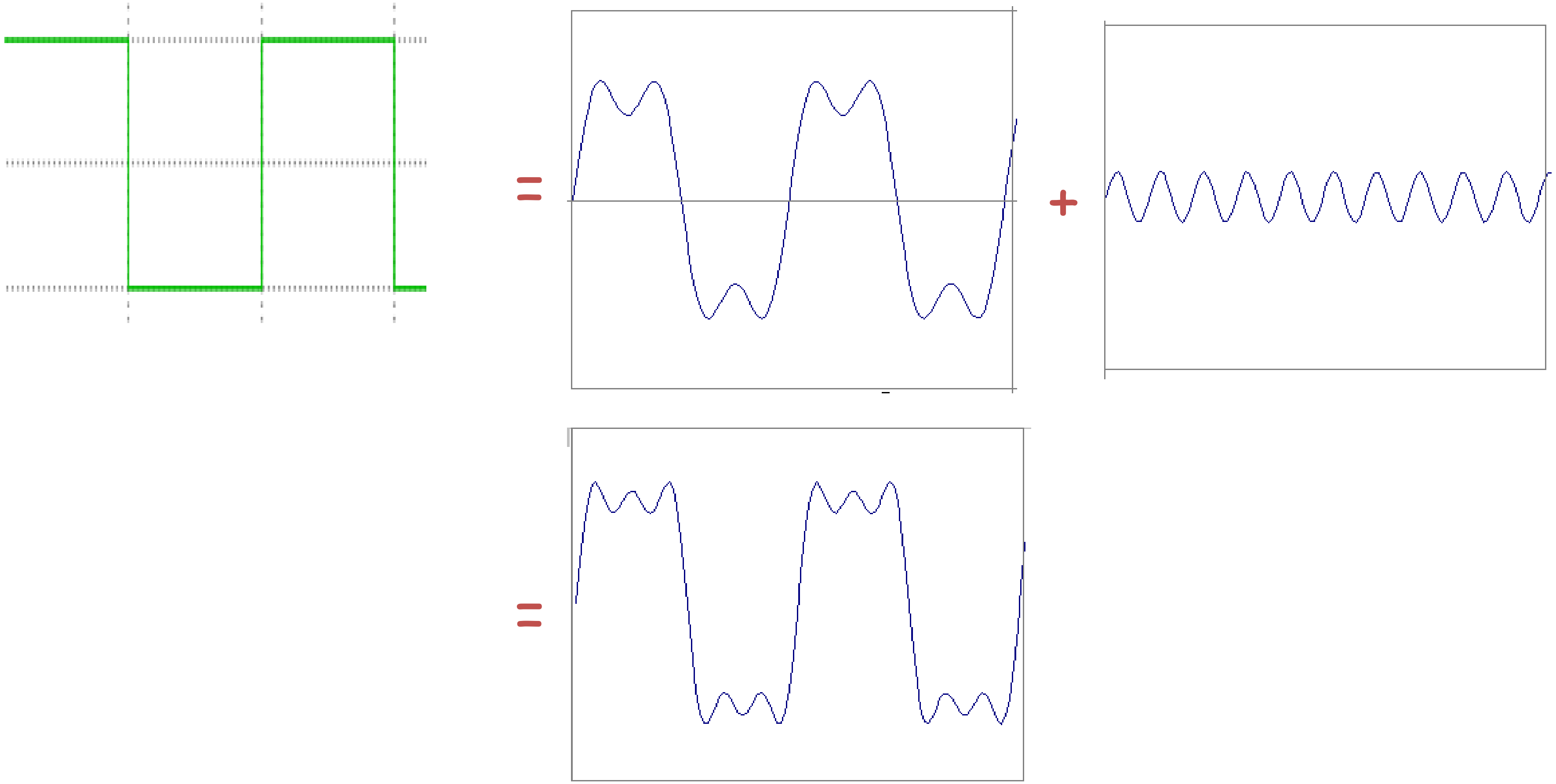
Frequency Spectra



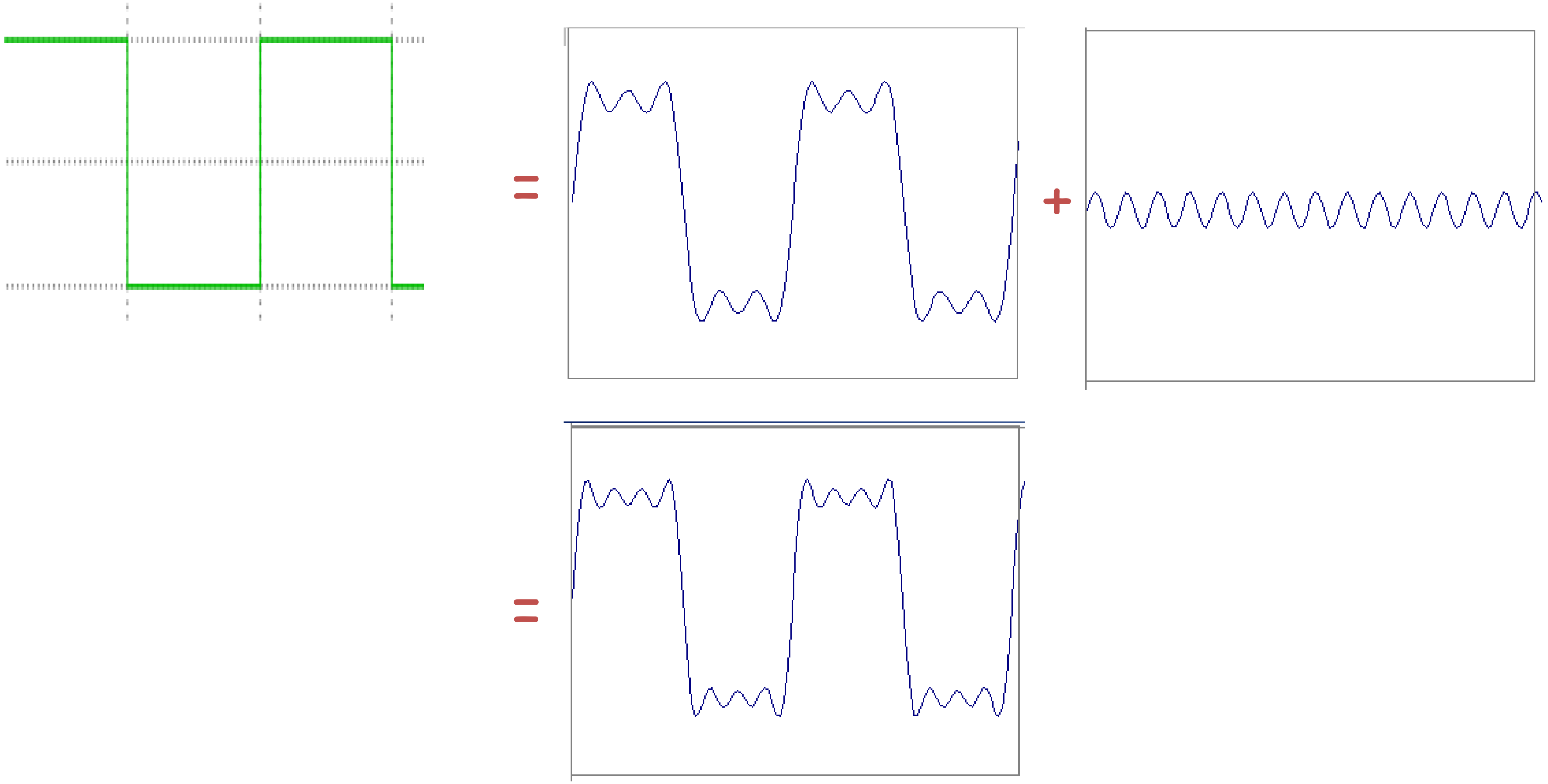
Frequency Spectra



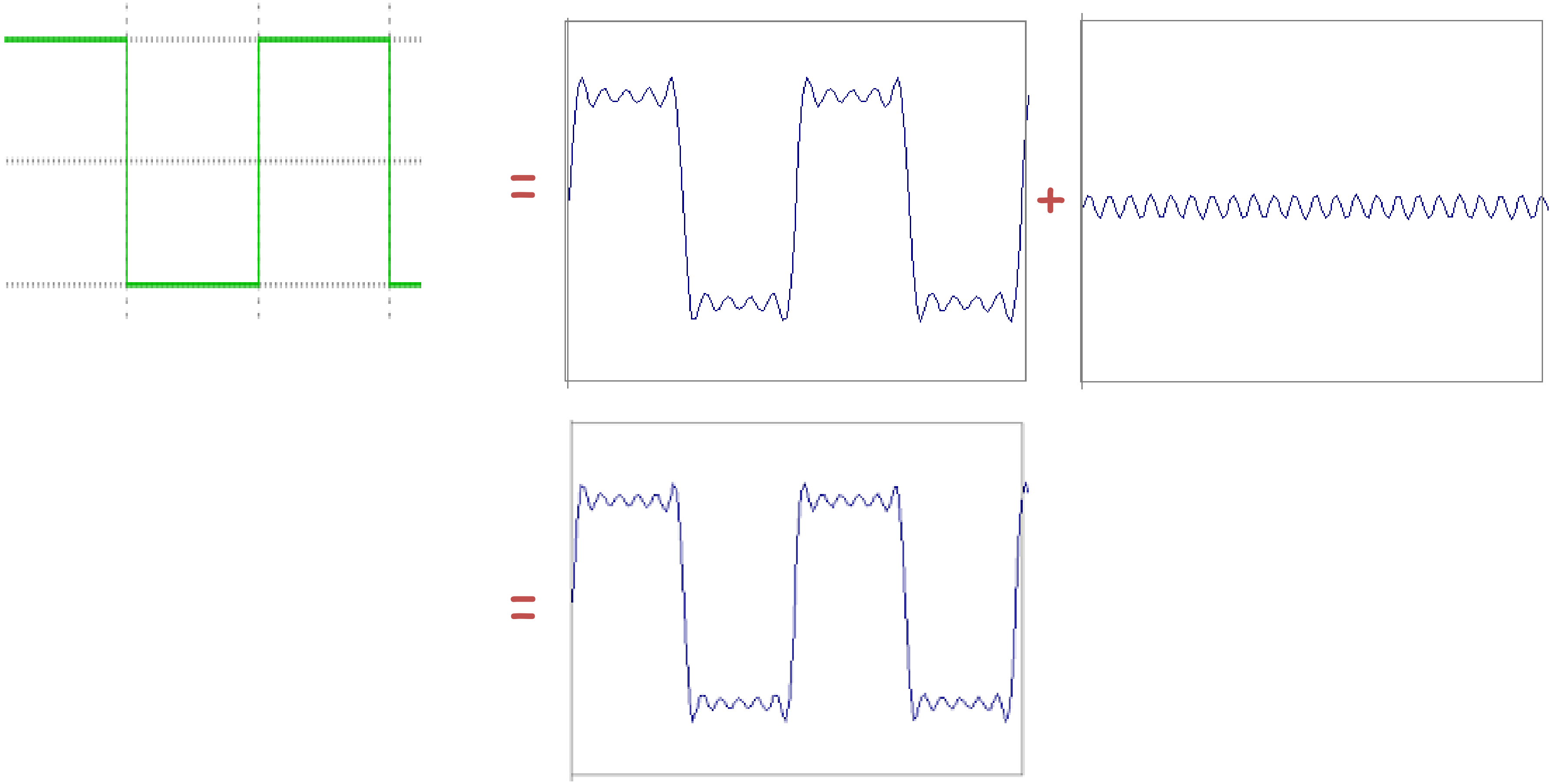
Frequency Spectra



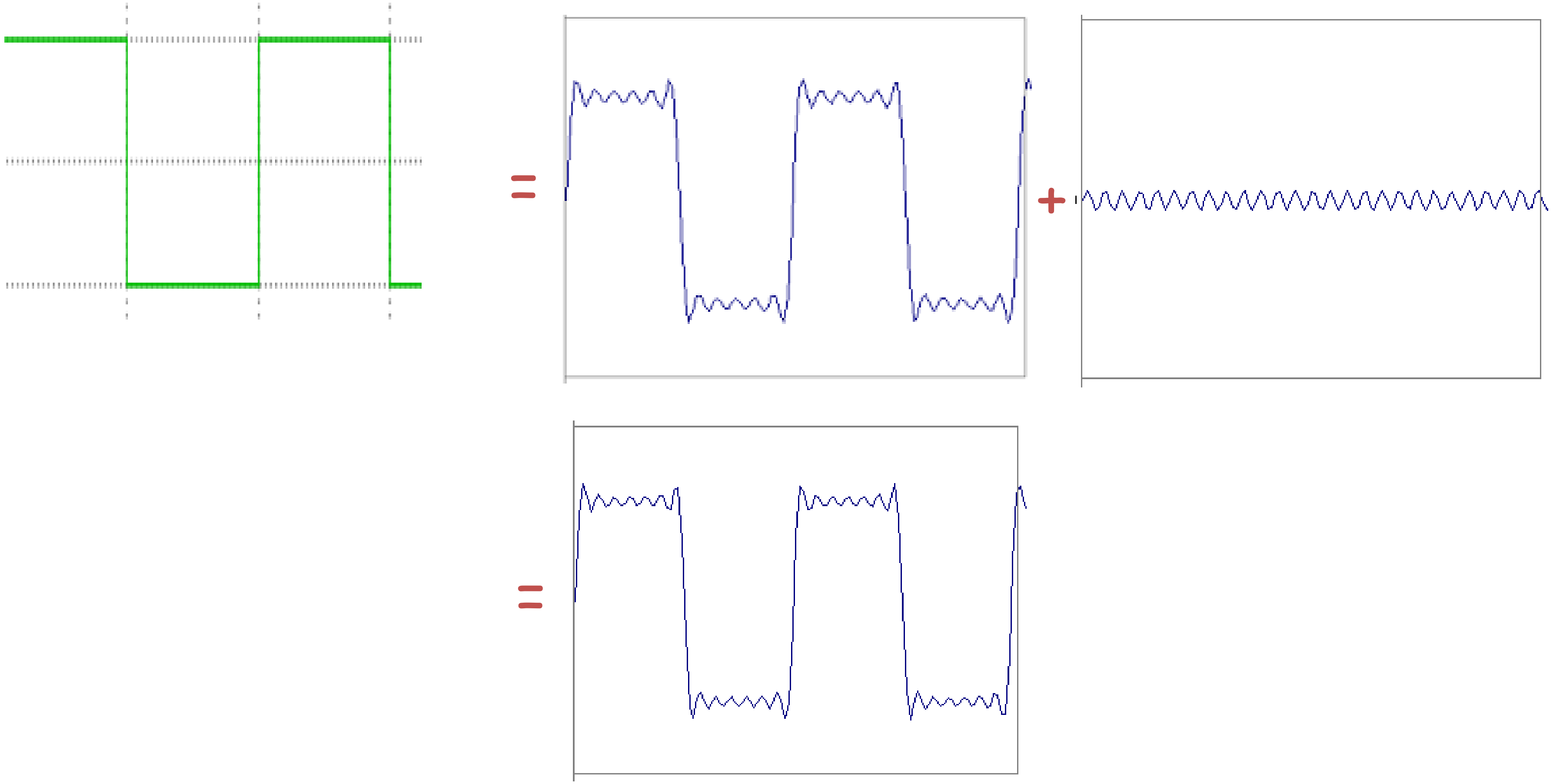
Frequency Spectra



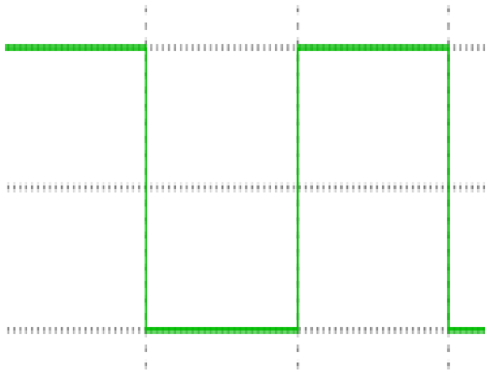
Frequency Spectra



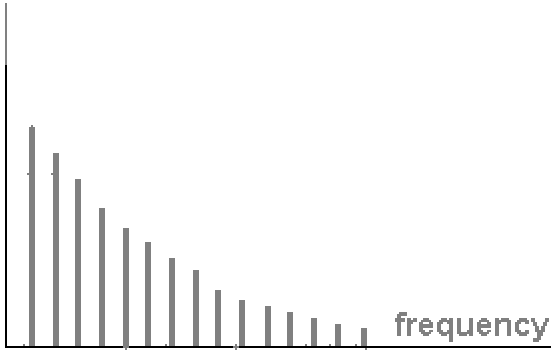
Frequency Spectra



Frequency Spectra

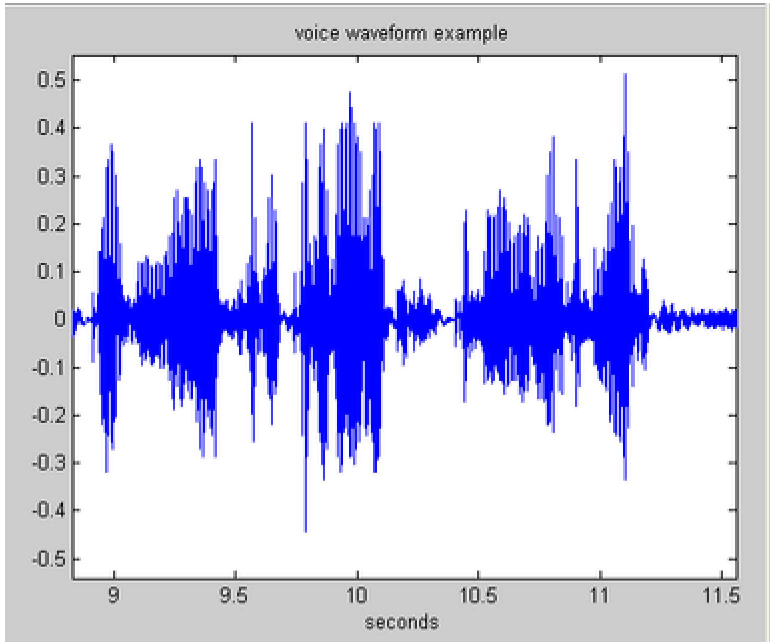


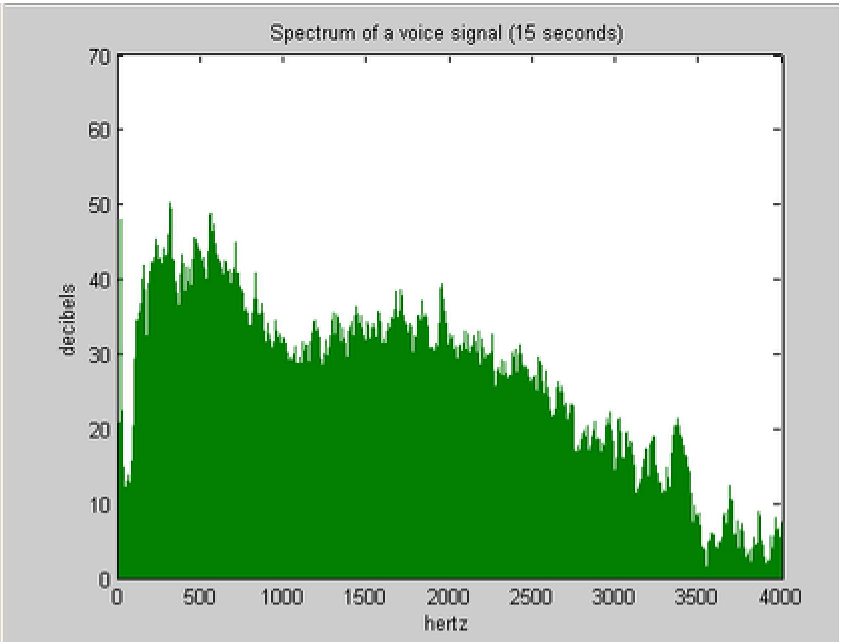
$$= \boxed{A\sum_{k=1}^{\infty}\frac{1}{k}\sin(2\pi kt)}$$



Example: Music

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

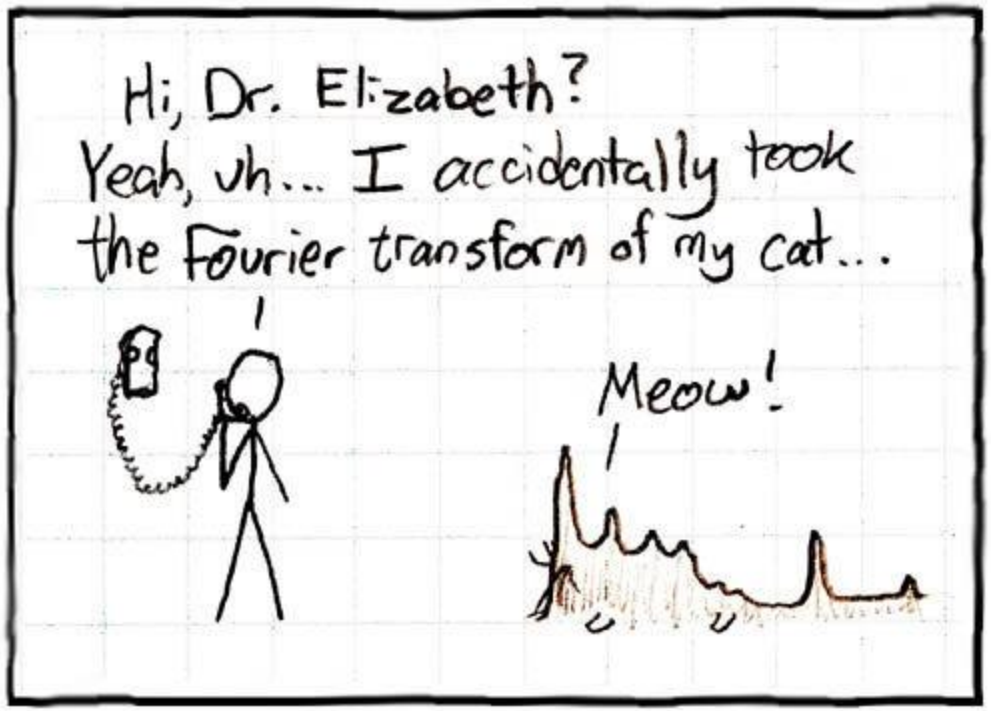




Slide: Hoiem

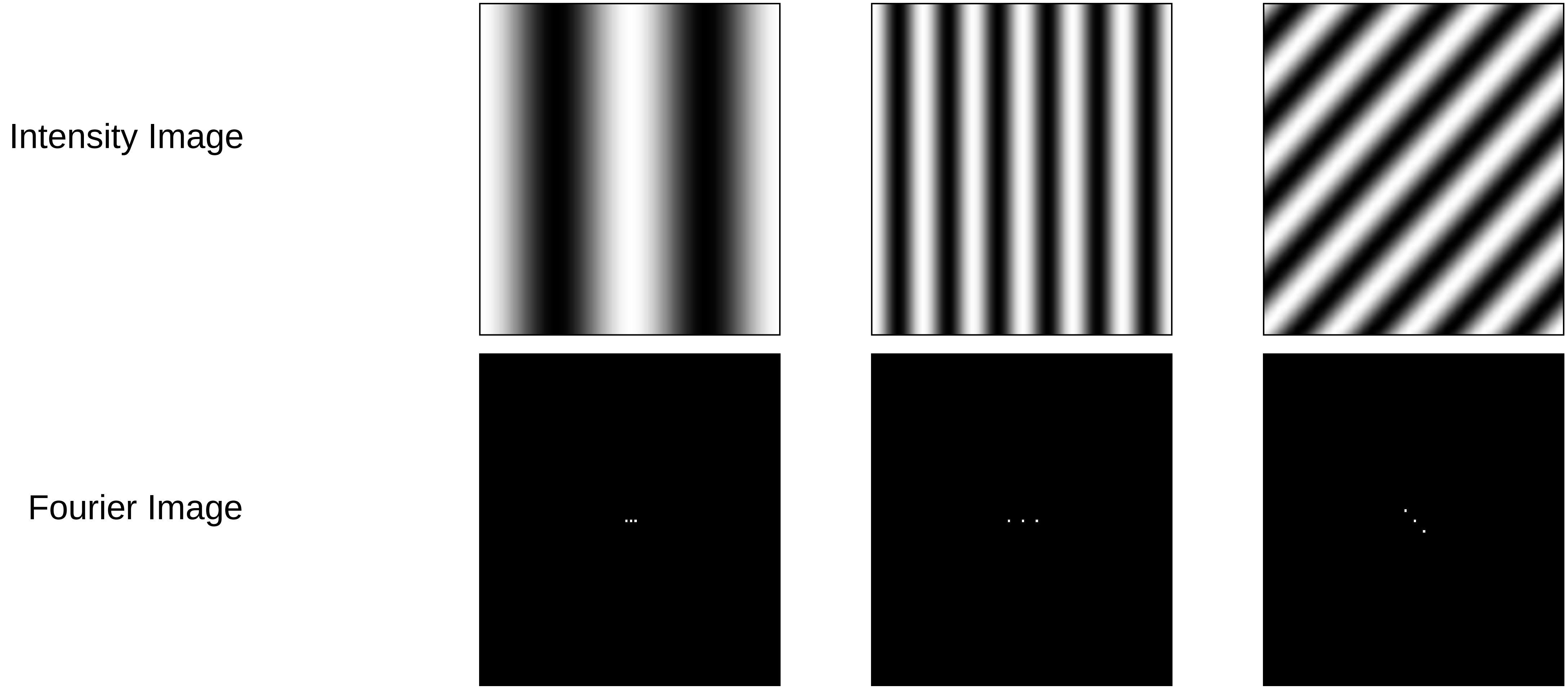
Other signals

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



xkcd.com

Fourier analysis in images



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

Fourier Transform

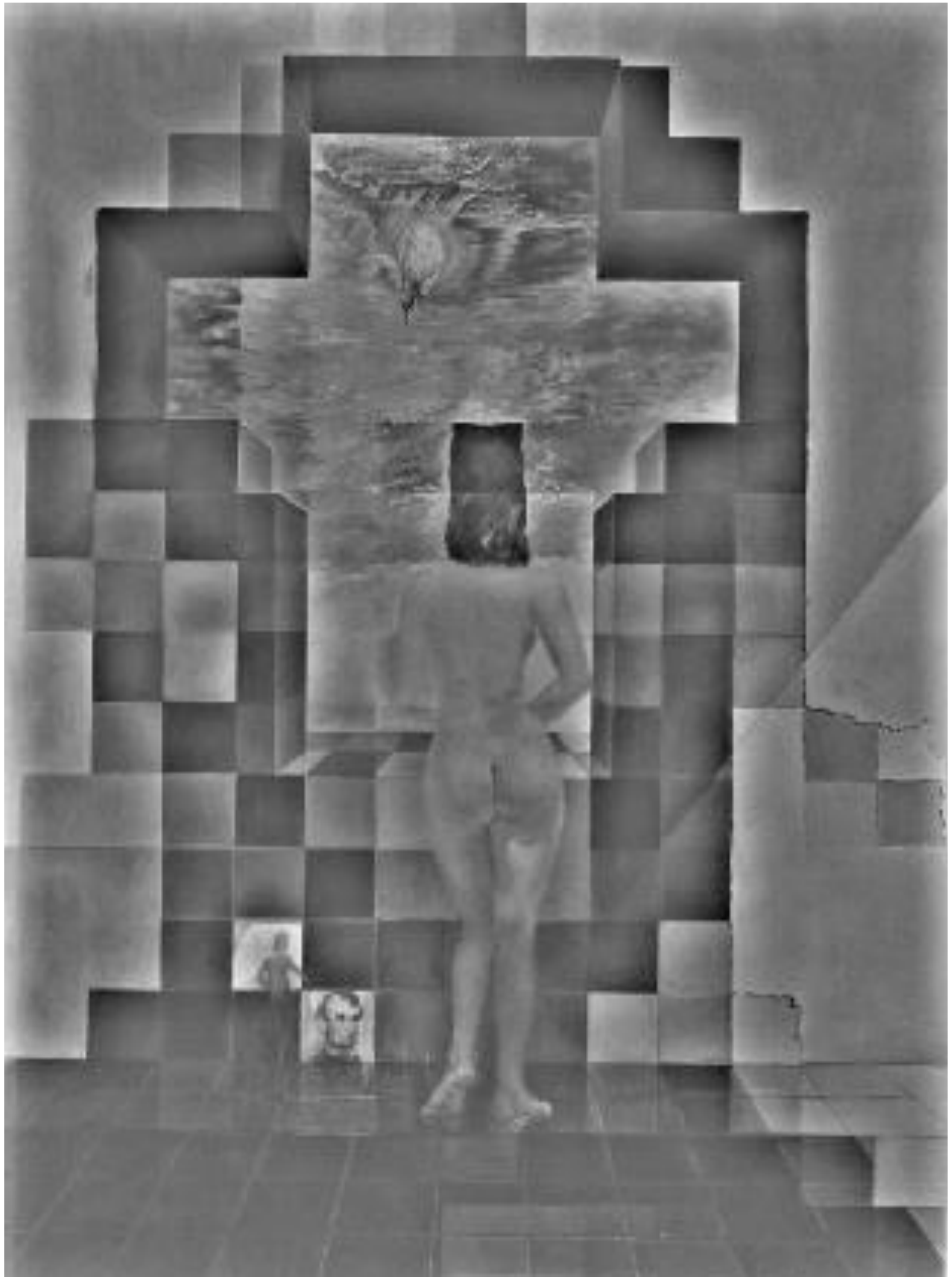
- Fourier transform stores the magnitude and phase at each frequency
  - Magnitude encodes how much signal there is at a particular frequency
  - Phase encodes spatial information (indirectly)
  - For mathematical convenience, this is often notated in terms of real and complex numbers

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

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

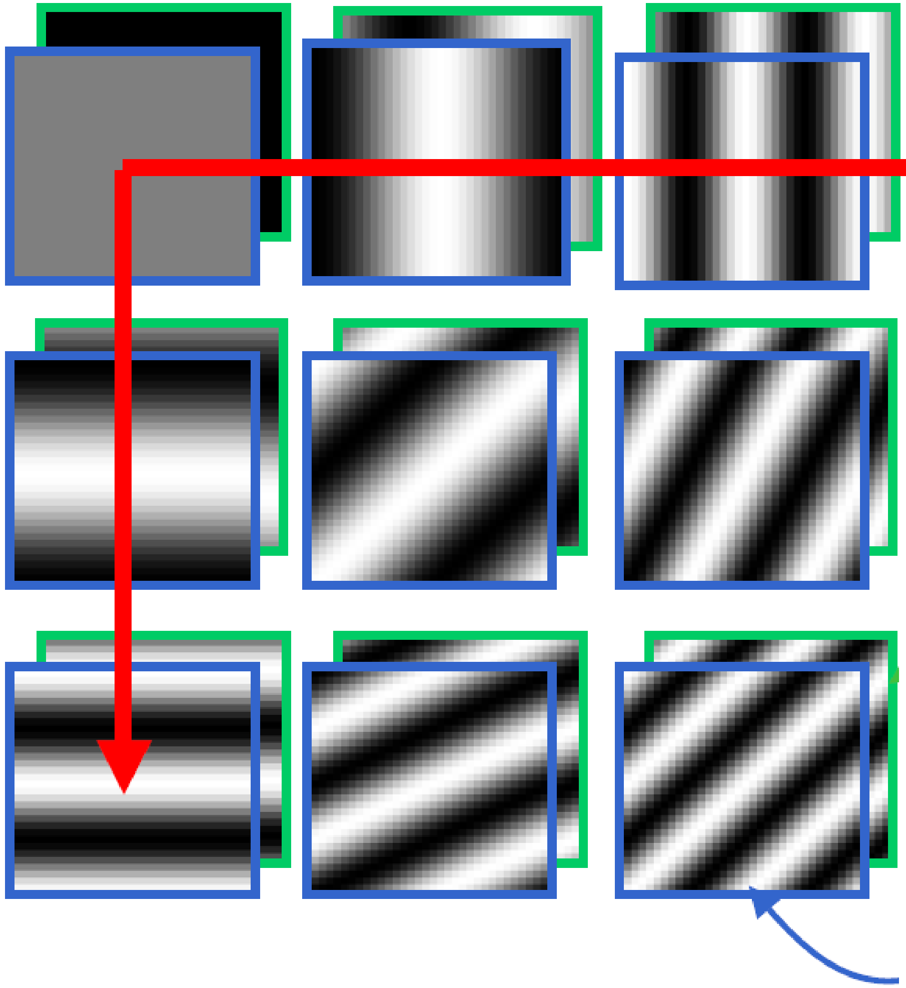
Salvador Dali invented Hybrid Images?

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



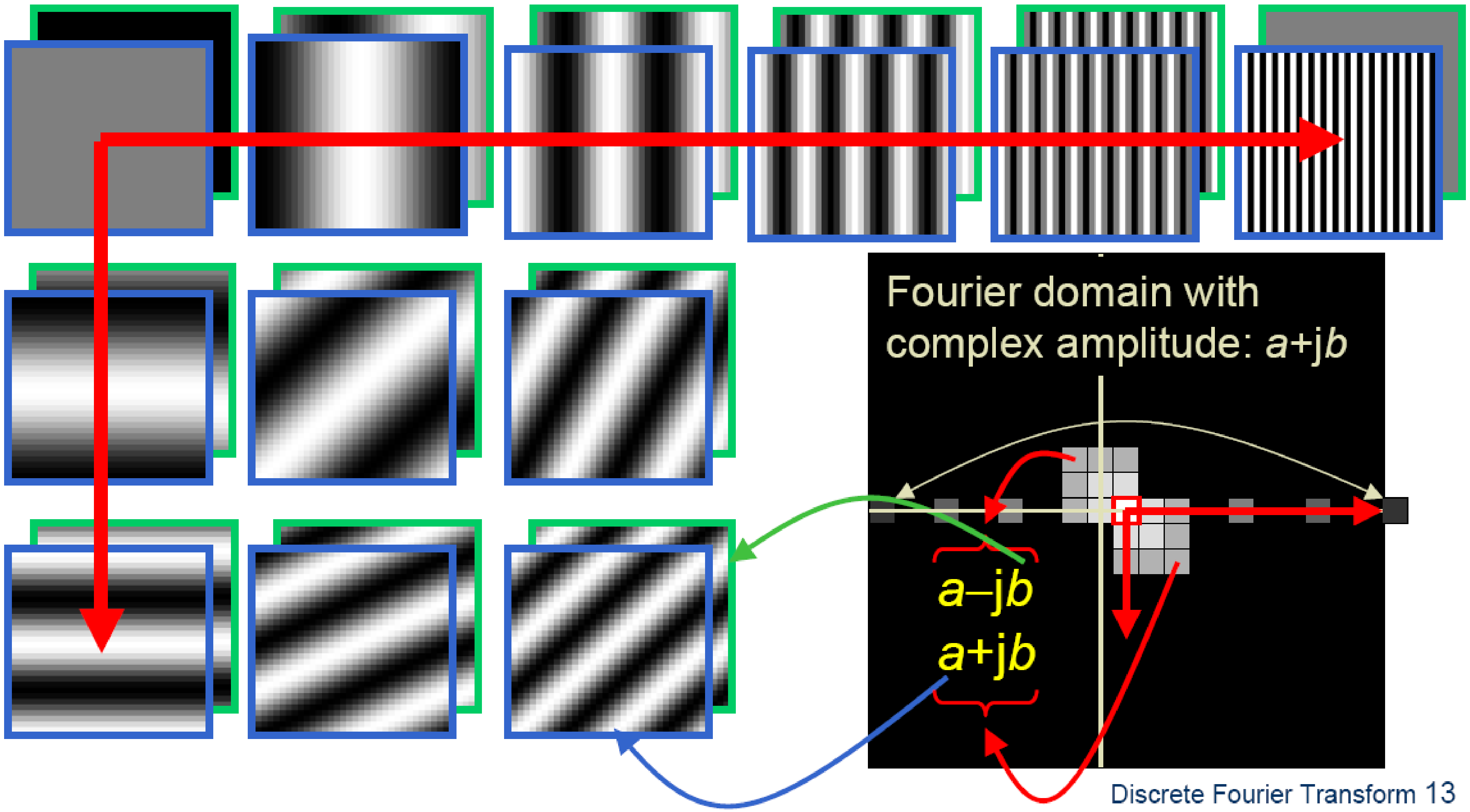
Fourier Bases

Teases away fast vs. slow changes in the image.

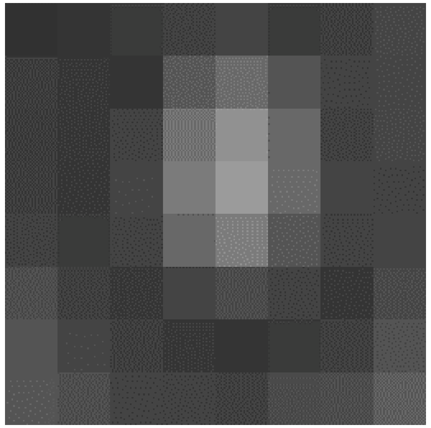


This change of basis is the Fourier Transform

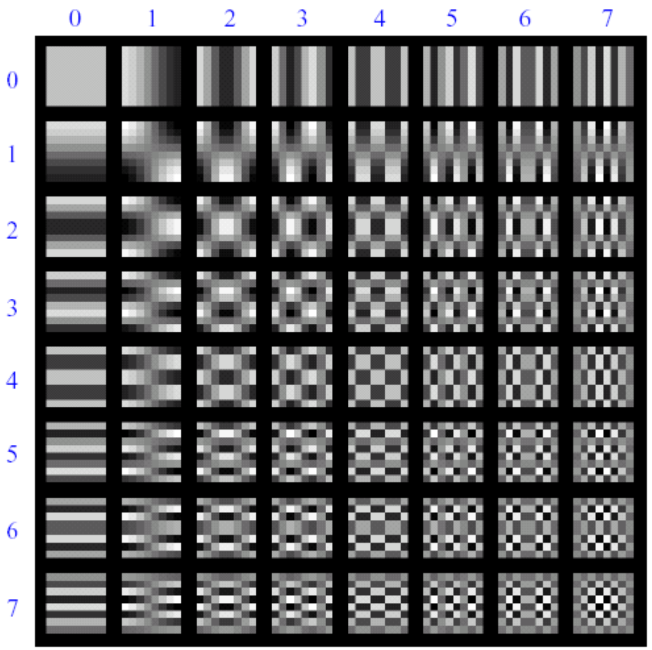
Fourier Bases



This looks a lot like DCT in JPEG compression



8x8 image patch

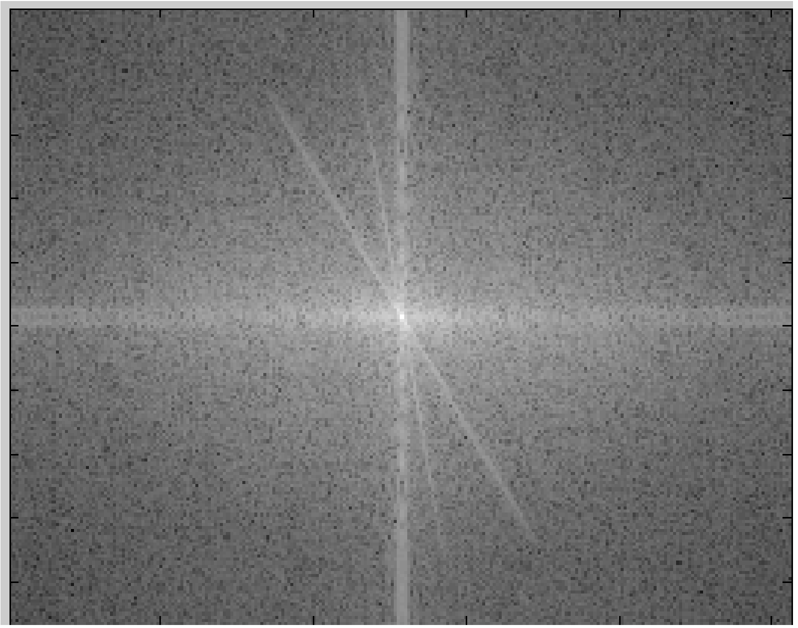


DCT bases

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

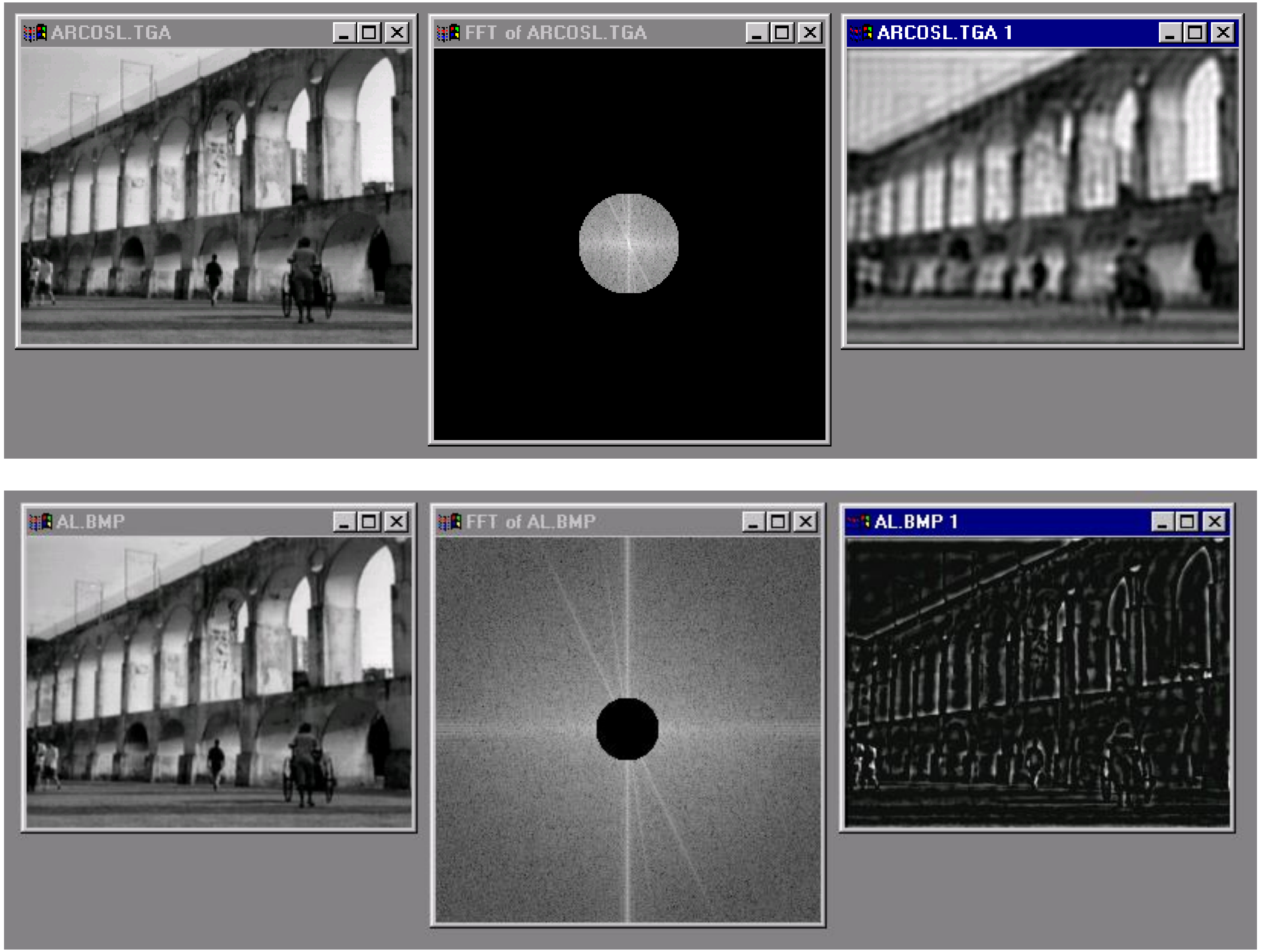
Patch representation after projecting on to DCT bases

Man-made Scene



Can change spectrum, then reconstruct

Low and High Pass filtering



Computing the Fourier Transform

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

Continuous

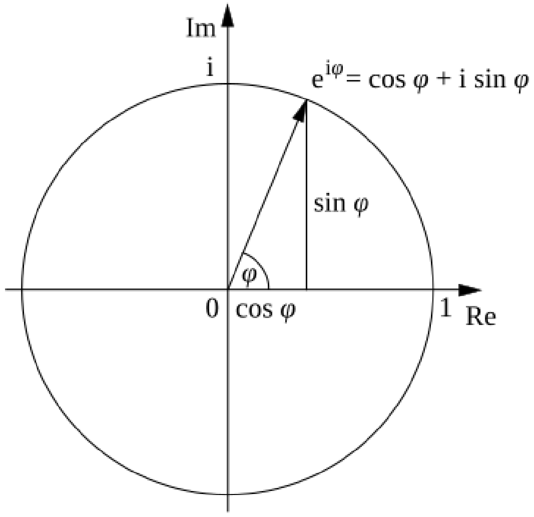
$$H(\omega) = \int_{-\infty}^{\infty} h(x)e^{-j\omega x}dx$$

Discrete

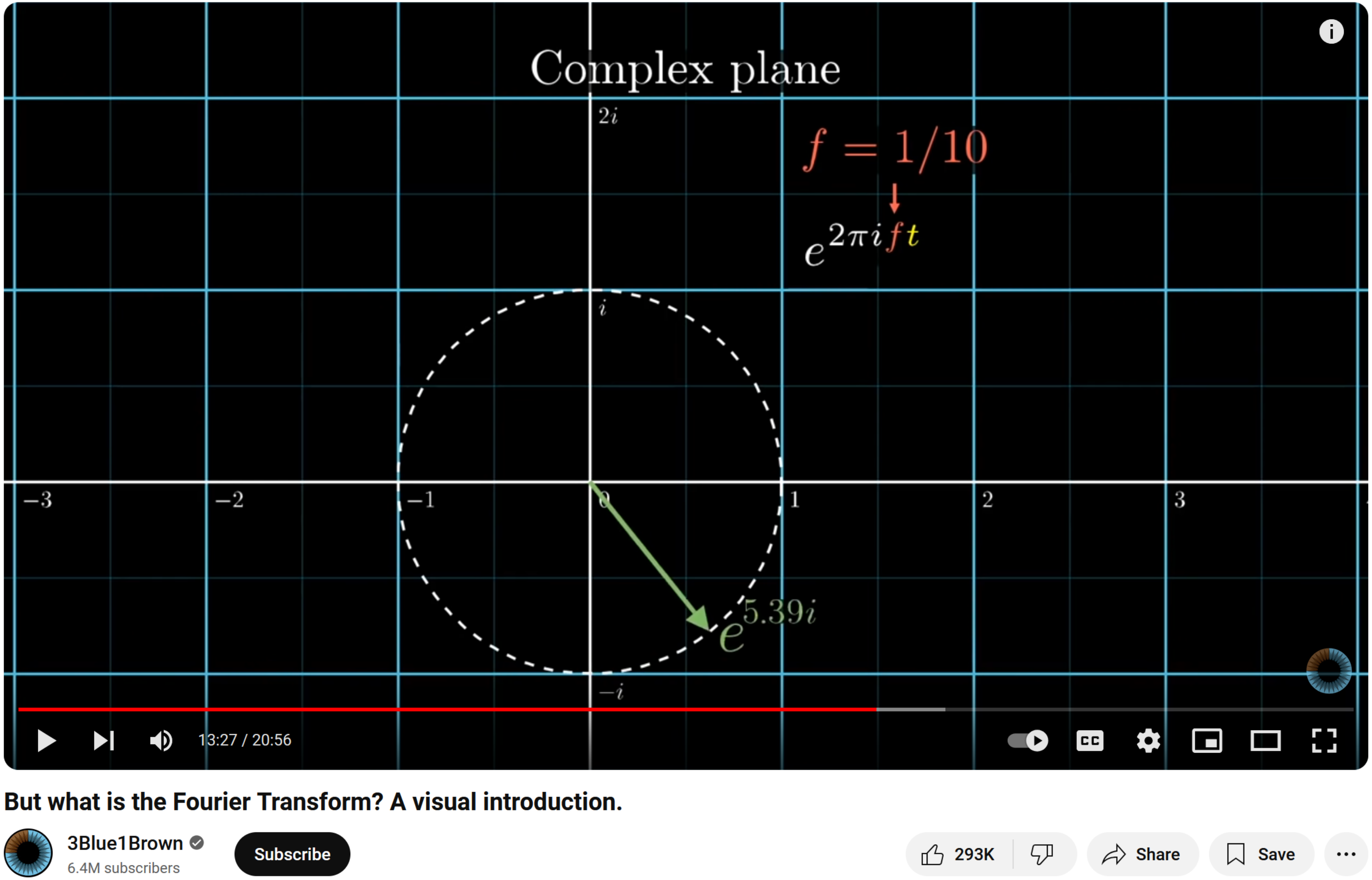
$$H(k) = \frac{1}{N}\sum_{x=0}^{N-1} h(x)e^{-j\frac{2\pi kx}{N}}$$

k = -N/2..N/2

Fast Fourier Transform (FFT): NlogN



Euler's Formula



https://youtu.be/spUNpyF58BY?si=93x8YxT5n45OA3CD

The Convolution Theorem

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

$$\mathrm{F}[g * h] = \mathrm{F}[g]\,\mathrm{F}[h]$$

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

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