Image compression and Fourier analysis

Vlad Shakhuro



1 October 2026

Outline

1. Lossless compression with PNG
2. Lossy compression with JPEG
3. Image analysis using Fourier transform
4. Image stitching with Laplacian pyramid

Lossless compression with PNG

PNG is a free alternative to GIF, proposed in 1995.

1. Supports only lossless compression.
2. RGBA, up to 16 bit per channel, supports palettes.
3. Simple cross-platform implementation.

Delta-coding

	C	B	D	
	A	X		

Coding methods:

0 Byte X is passed unaltered

1 Code $X - A$

2 Code $X - B$

3 Code $X - \left\lfloor \frac{A+B}{2} \right\rfloor$

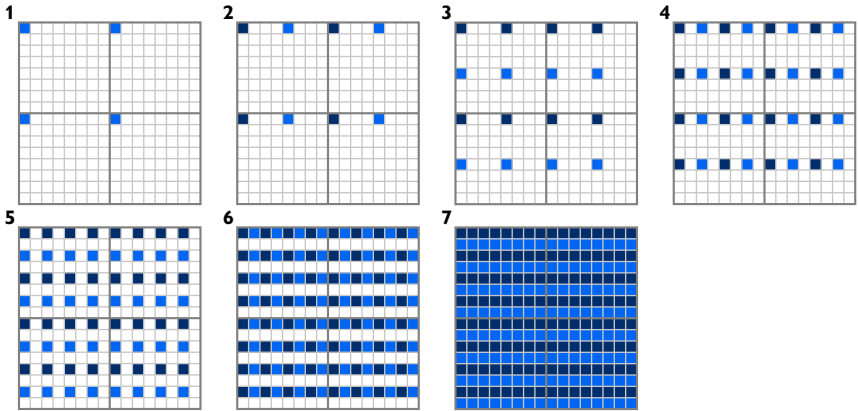
4 Code $X - F$, $F \in \{A, B, C\}$, nearest to $p = A + B - C$

Coding method is chosen per line

Obtained sequence of numbers is compressed using DEFLATE algorithm (used in ZIP)

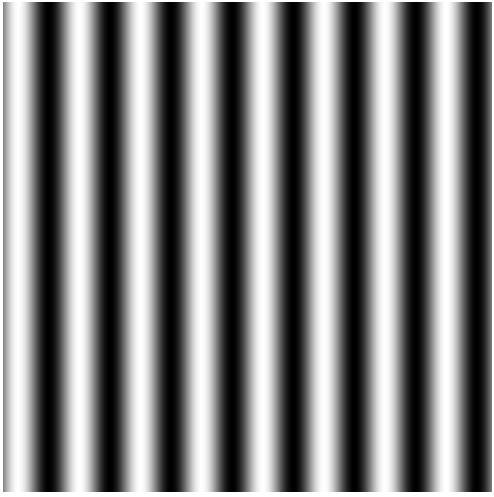
Image traversal (Adam7)

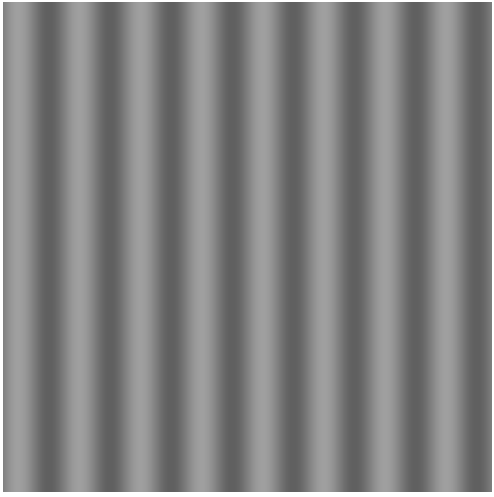
PNG can optionally use traversal in 7 steps for partial transmission of an image

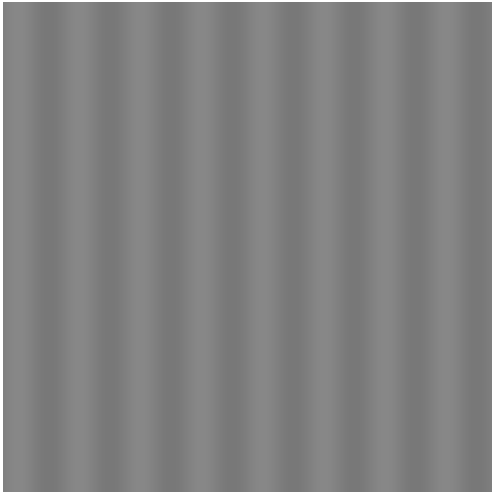


Outline

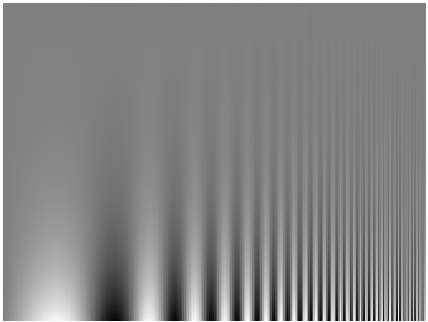
1. Lossless compression with PNG
2. Lossy compression with JPEG
3. Image analysis using Fourier transform
4. Image stitching with Laplacian pyramid

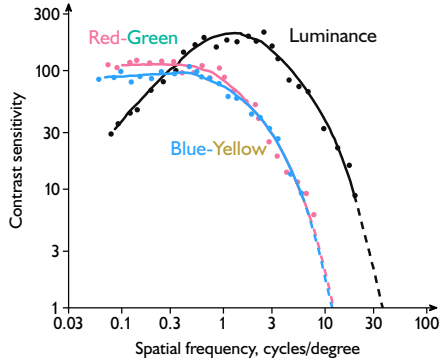






Contrast Sensitivity Function





Humans perceive luminance and color with different resolutions. To make use of this fact, image transmission systems encode separate luminance component

Mullen. The Contrast Sensitivity of Human Colour Vision. 1985

Human sensitivity to luminance and color



RGB



luminance



chrominance

Human sensitivity to luminance and color



RGB



blurred luminance



blurred chrominance

Convert RGB to YC_bC_r

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$
$$C_b = 128 + 0.564 \cdot (B - Y)$$
$$C_r = 128 + 0.713 \cdot (R - Y)$$



Subsample chroma

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$
$$C_b = 128 + 0.564 \cdot (B - Y)$$
$$C_r = 128 + 0.713 \cdot (R - Y)$$









Shift by 128

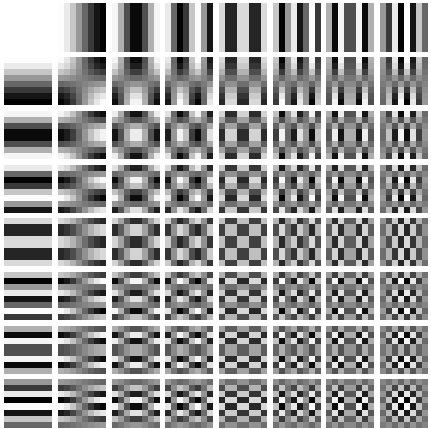
$$\begin{bmatrix} 139 & 144 & 149 & 153 & 155 & 155 & 155 & 155 \\ 144 & 151 & 153 & 156 & 159 & 156 & 156 & 156 \\ 150 & 155 & 160 & 163 & 158 & 156 & 156 & 156 \\ 159 & 161 & 162 & 160 & 160 & 159 & 159 & 159 \\ 159 & 160 & 161 & 162 & 162 & 155 & 155 & 155 \\ 161 & 161 & 161 & 161 & 160 & 157 & 157 & 157 \\ 162 & 162 & 161 & 163 & 162 & 157 & 157 & 157 \\ 162 & 162 & 162 & 161 & 163 & 158 & 158 & 158 \end{bmatrix}$$

source image

$$\begin{bmatrix} 11 & 16 & 21 & 25 & 27 & 27 & 27 & 27 \\ 16 & 23 & 25 & 28 & 31 & 28 & 28 & 28 \\ 22 & 27 & 32 & 35 & 30 & 28 & 28 & 28 \\ 31 & 33 & 34 & 32 & 32 & 31 & 31 & 31 \\ 31 & 32 & 33 & 34 & 34 & 27 & 27 & 27 \\ 33 & 33 & 33 & 33 & 32 & 29 & 29 & 29 \\ 34 & 34 & 33 & 35 & 34 & 29 & 29 & 29 \\ 34 & 34 & 34 & 33 & 35 & 30 & 30 & 30 \end{bmatrix}$$

shifted image

Discrete cosine transform



x

$$\begin{bmatrix} 11 & 16 & 21 & 25 & 27 & 27 & 27 & 27 \\ 16 & 23 & 25 & 28 & 31 & 28 & 28 & 28 \\ 22 & 27 & 32 & 35 & 30 & 28 & 28 & 28 \\ 31 & 33 & 34 & 32 & 32 & 31 & 31 & 31 \\ 31 & 32 & 33 & 34 & 34 & 27 & 27 & 27 \\ 33 & 33 & 33 & 33 & 32 & 29 & 29 & 29 \\ 34 & 34 & 33 & 35 & 34 & 29 & 29 & 29 \\ 34 & 34 & 34 & 33 & 35 & 30 & 30 & 30 \end{bmatrix}$$

y

u

$$\begin{bmatrix} 235.6 & -1.0 & -12.1 & -5.2 & 2.1 & -1.7 & -2.7 & 1.3 \\ -22.6 & -17.5 & -6.2 & -3.2 & -2.9 & -0.1 & 0.4 & -1.2 \\ -10.9 & -9.3 & -1.6 & 1.5 & 0.2 & -0.9 & -0.6 & -0.1 \\ -7.1 & -1.9 & 0.2 & 1.5 & 0.9 & -0.1 & 0.0 & 0.3 \\ -0.6 & -0.8 & 1.5 & 1.6 & -0.1 & -0.7 & 0.6 & 1.3 \\ 1.8 & -0.2 & 1.6 & -0.3 & -0.8 & 1.5 & 1.0 & -1.0 \\ -1.3 & -0.4 & -0.3 & -1.5 & -0.5 & 1.7 & 1.1 & -0.8 \\ -2.6 & 1.6 & -3.8 & -1.8 & 1.9 & 1.2 & -0.6 & -0.4 \end{bmatrix}$$

v

amplitudes

Amplitude quantization

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

luminance quantization matrix

$$\begin{bmatrix} 15 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

quantized amplitudes

Quantization matrices

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

luminance quantization matrix

$$\begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix}$$

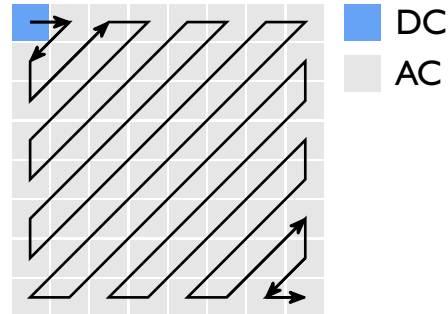
color quantization matrix

Quantization matrix depends on quality factor Q (from 1 to 100) set by user. Quantization matrices are constructed using:

1. Amplitude statistics of real images.
2. Model for human visual system.

Zigzag scanning

$$\begin{bmatrix} 15 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$




```
15 0 -2 -1 -1 -1 0 0 -1 0 ... 0
```

Sequence of numbers is compressed using Zero Run-Length Encoding and common compression method (Huffman or arithmetic coding)

DC (direct current): mean of block, AC (alternating current): oscillating components

JPEG algorithm

1. Convert image in color space YC_bC_r.
2. Downsample color components.
3. Divide image into blocks of size 8×8 pixels, subtract 128.
4. For every block:
    1. Apply discrete cosine transform.
    2. Quantize coefficients (amplitudes).
    3. Compress quantized coefficients.

Algorithm specificity:

1. Designed for compressing images that are shown on a screen.
2. Images are compressed due to loss of high-frequency information.

Progressive JPEG

Baseline JPEG sends blocks top to bottom. Progressive JPEG sends whole image in several scans, each refining previous one:

- spectral selection: DC first, then zigzag AC bands
- successive approximation: high bits first, low bits later

1	0	1	0	0	= 20
1	0	1	1	0	= 22
1	0	1	1	1	= 23



scan 1: DC
11% of file



scan 2: AC 1–5
22% of file



scan 5: AC 6–63
35% of file



scan 10: low bits
100% of file

Outline

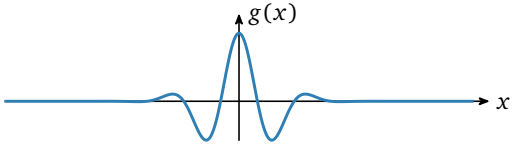
1. Lossless compression with PNG
2. Lossy compression with JPEG
3. Image analysis using Fourier transform
4. Image stitching with Laplacian pyramid

Fourier transform

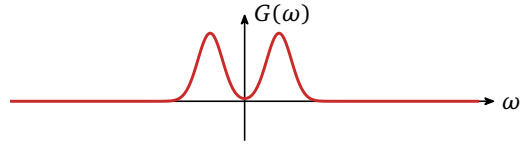
$$\underbrace{G(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} g(x) e^{-i\omega x} dx}_{\text{forward Fourier transform, } \mathscr{F}}$$

$$\underbrace{g(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} G(\omega) e^{i\omega x} d\omega}_{\text{inverse Fourier transform, } \mathscr{F}^{-1}}$$

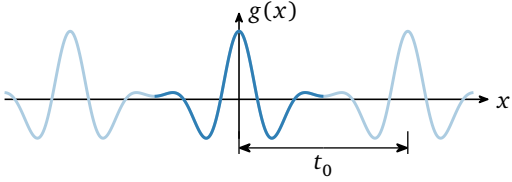
Function and spectrum properties



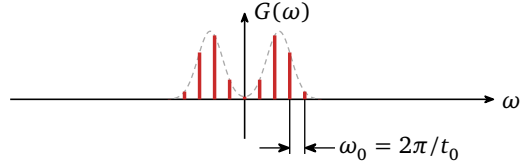
Continuous aperiodic signal



Continuous aperiodic spectrum



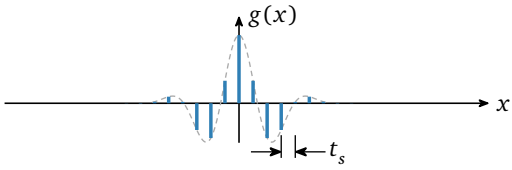
Continuous periodic signal



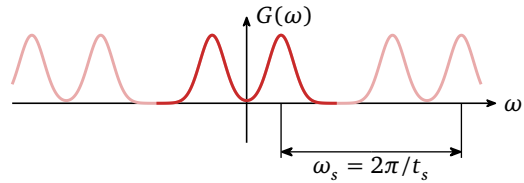
Discrete spectrum

Periodic $g \Rightarrow$ discrete G: Fourier series

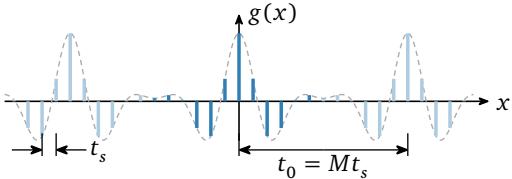
Function and spectrum properties



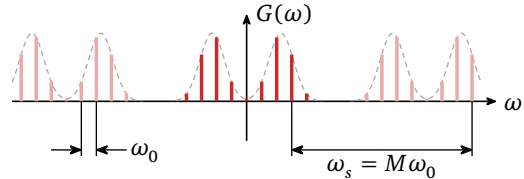
Discrete signal



Periodic spectrum



Discrete periodic signal



Discrete periodic spectrum

Discrete $g \Rightarrow$ periodic G: Poisson summation formula

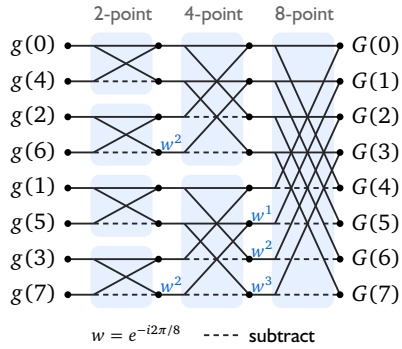
Discrete Fourier Transform

$$G(m) = \frac{1}{\sqrt{M}} \sum_{u=0}^{M-1} g(u) e^{-i2\pi \frac{mu}{M}}, \quad 0 \le m < M$$

$$g(u) = \frac{1}{\sqrt{M}} \sum_{m=0}^{M-1} G(m) e^{i2\pi \frac{mu}{M}}, \quad 0 \le u < M$$

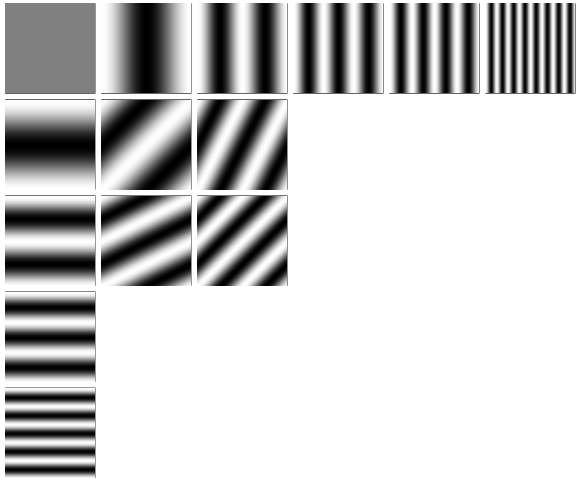
Naive implementation: $O(M^2)$

Fast implementation: $O(M \log M)$



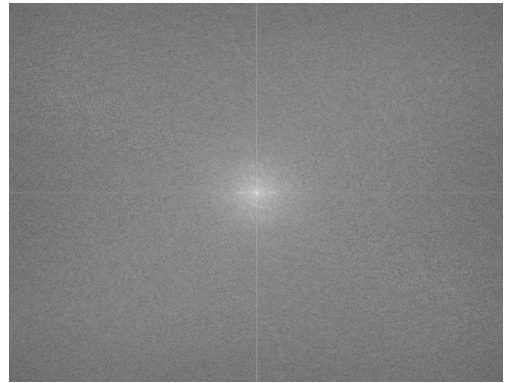
FFT(M) splits into two FFT($M/2$) over even and odd samples, combined in $O(M)$. Recursion depth $\log_2 M$ gives $O(M \log M)$.

Fourier basis for images





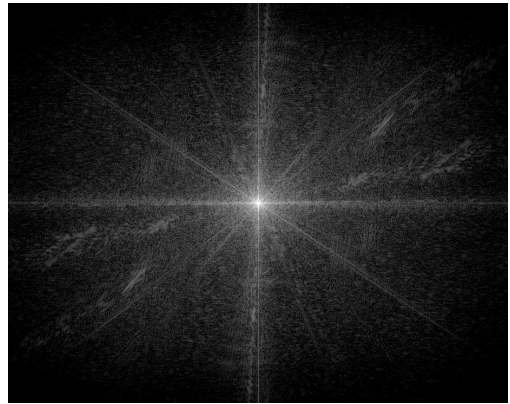
spatial domain



frequency domain

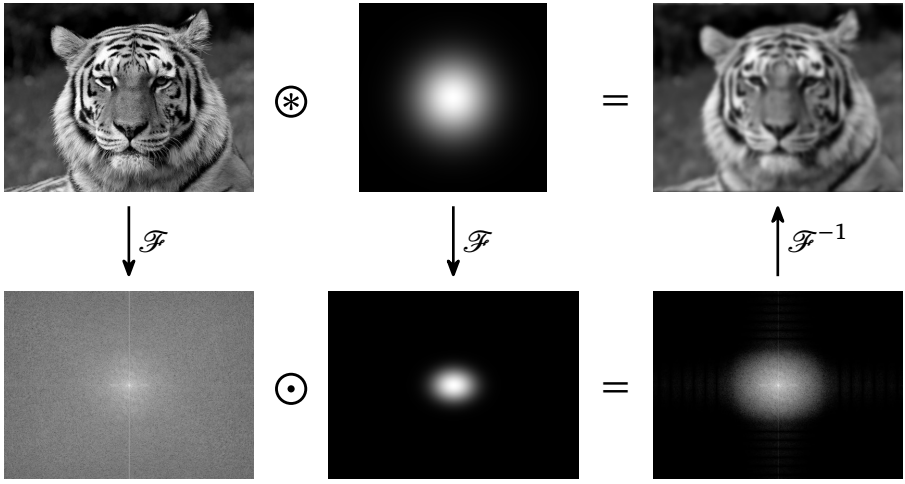
```
freq = log(1 + abs(fftshift(fft2(img))))
```



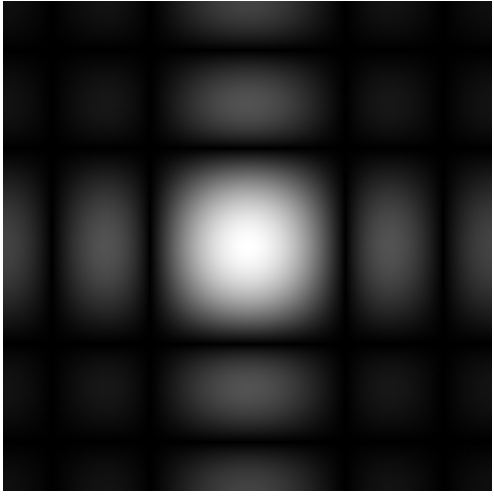




Convolution theorem



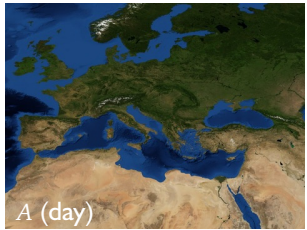
Fourier image of box filter



Outline

1. Lossless compression with PNG
2. Lossy compression with JPEG
3. Image analysis using Fourier transform
4. Image stitching with Laplacian pyramid

Image stitching



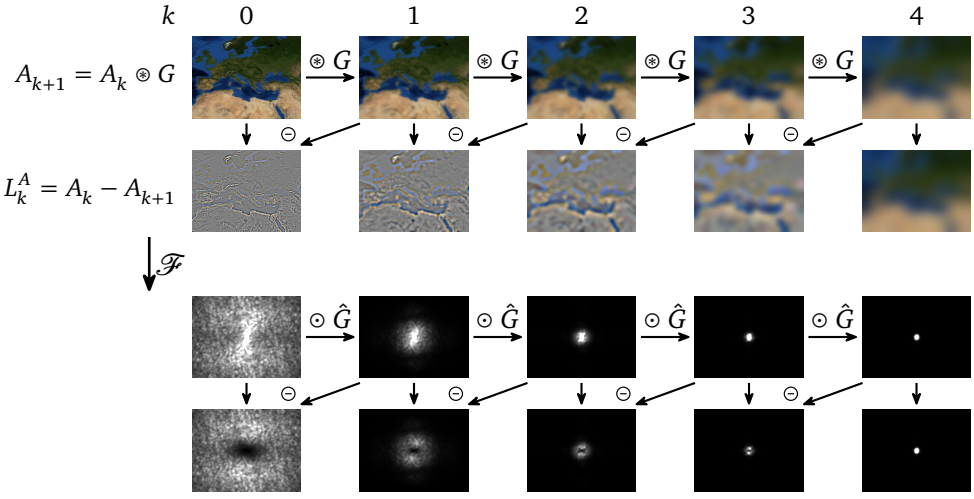




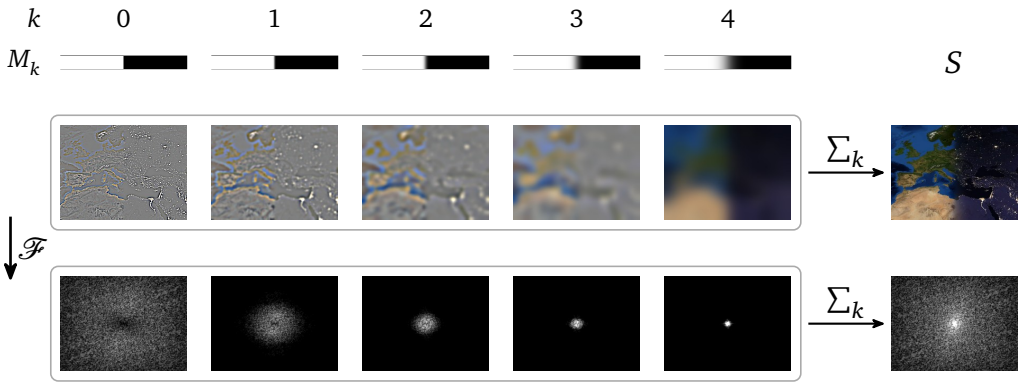




Building Laplacian pyramid



Blending Laplacian pyramids



$$L_k^S = M_k \odot L_k^A + (1 - M_k) \odot L_k^B$$

Laplacian pyramid blending

Input: images A, B and mask M (1 where A is taken)

1. Build Gaussian pyramids: $A_{k+1} = A_k \circledast G$, same for B, M
2. Build Laplacian pyramids: $L_k^A = A_k - A_{k+1}$, $L_N^A = A_N$, same for B
3. Blend each level: $L_k^S = M_k \odot L_k^A + (1 - M_k) \odot L_k^B$
4. Collapse into blended image: $S = \sum_k L_k^S$

Burt, Adelson. A Multiresolution Spline With Application to Image Mosaics. 1983

Conclusion

We reviewed following topics:

- PNG lossless image compression. It uses simple delta coding and DEFLATE compression for RGBA pixel values
- JPEG lossy image compression. It is based on discrete cosine transform, quantization and compression of amplitudes
- Fourier transform — a useful tool for analysis of images and linear filters
- Image stitching via Laplacian pyramid. We may justify its applicability using Fourier analysis