Digital Communications
SSY125, Lecture 4

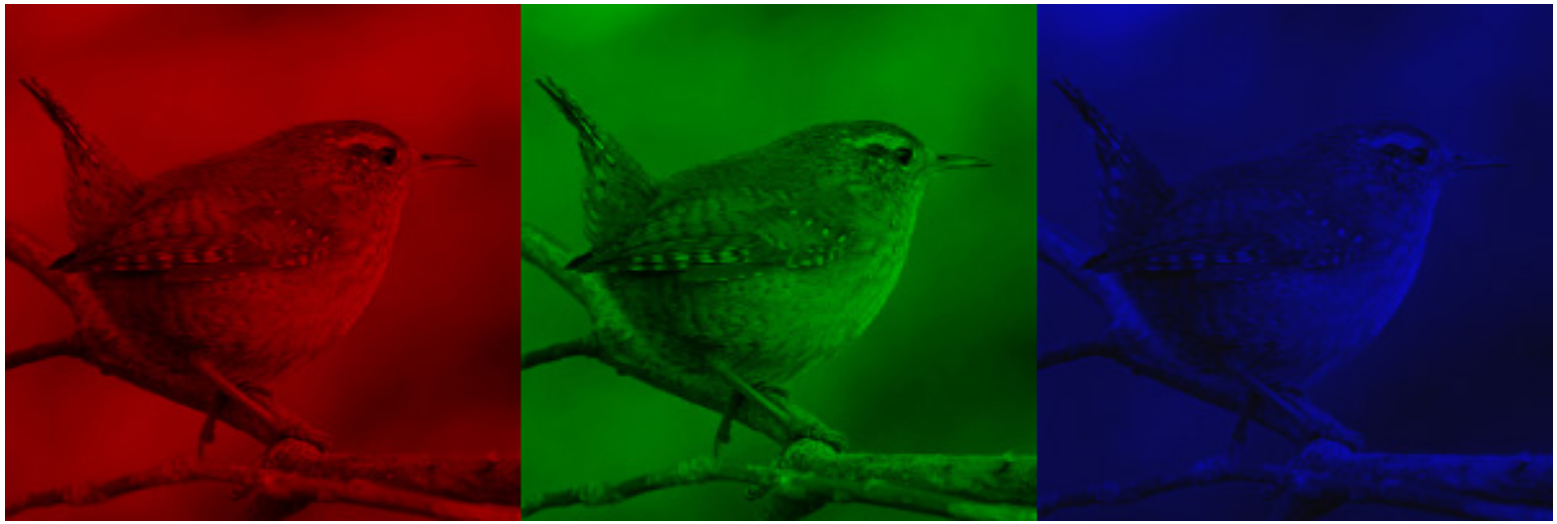
JPEG compression

Alexandre Graell i Amat
alexandre.graell@chalmers.se
https://sites.google.com/site/agraellamat

CHALMERS

November 11, 2019

A digital image



- An image consists of pixels
- Typically, natural photographic images are represented in the RGB color space, where R, G, and B, are the red, green, and blue component of a pixel
- Each pixel is typically represented with 8 bits for red, green, and blue colors, i.e., a total of 24 bits

JPEG: principle

- JPEG compresses the image by discarding information the human eye cannot easily see
- The human eye barely notices slight changes in color but is very sensitive to slight changes in brightness or contrast $\longrightarrow$ JPEG lossy compression is aggressive with the color component of the image
- A typical photographic quality image maybe compressed by 20:1 without experiencing any noticeable degradation in quality

JPEG steps:

1. Separate luminance from color $\longrightarrow$ JPEG first converts RGB images to the YCbCr color space

Y: luma (luminance) component of the color
Cb: blue chroma (color) component
Cr: red chroma (color) component

JPEG: principle

8 bits per pixel

JPEG: downsampling

JPEG steps:

1. Separate luminance from color $\longrightarrow$ JPEG first converts RGB images to the YCbCr color space
2. Apply downsampling to the chroma components, i.e., reduce its resolution

Simple way of exploiting human eye's lesser sensitivity to color information:

- Use less pixels for the two chroma components $\longrightarrow$ downsampling
- Luma component retained at full resolution

Chroma components are typically downsampled by 2:1 horizontally and either 1:1 or 2:1 vertically

For a color image of 1000×1000 pixels: luminance channel at 1000×1000 pixels, chroma channels either 500×1000 pixels or 500×500 pixels.

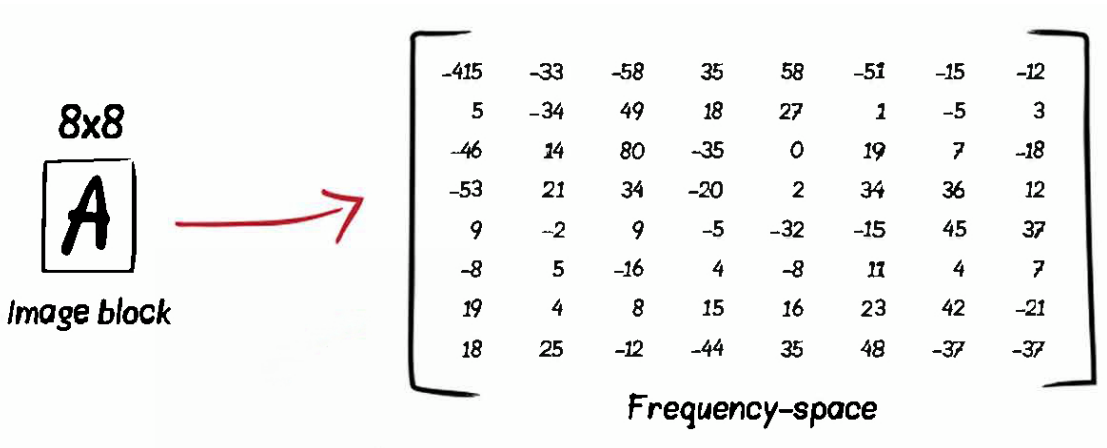
JPEG: DCT

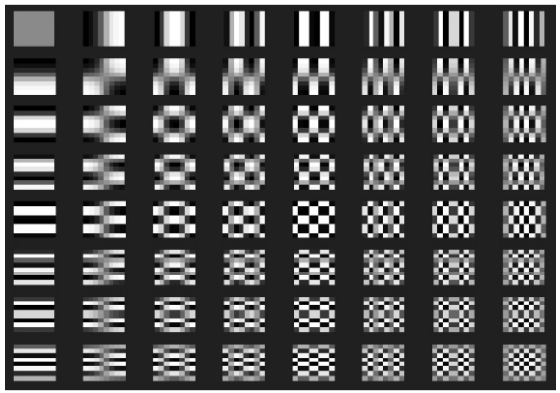
- JPEG works on blocks of 8×8 pixels, treating luminance and chroma components separately

JPEG steps:

1. Separate luminance from color $\longrightarrow$ JPEG first converts RGB images to the YCbCr color space
2. Apply downsampling to the chroma components, i.e., reduce its resolution
3. Each 8×8 block is converted to a frequency-domain representation by means of a discrete cosine transform (DCT) $\longrightarrow$ 8×8 block of frequency coefficient values

JPEG: DCT





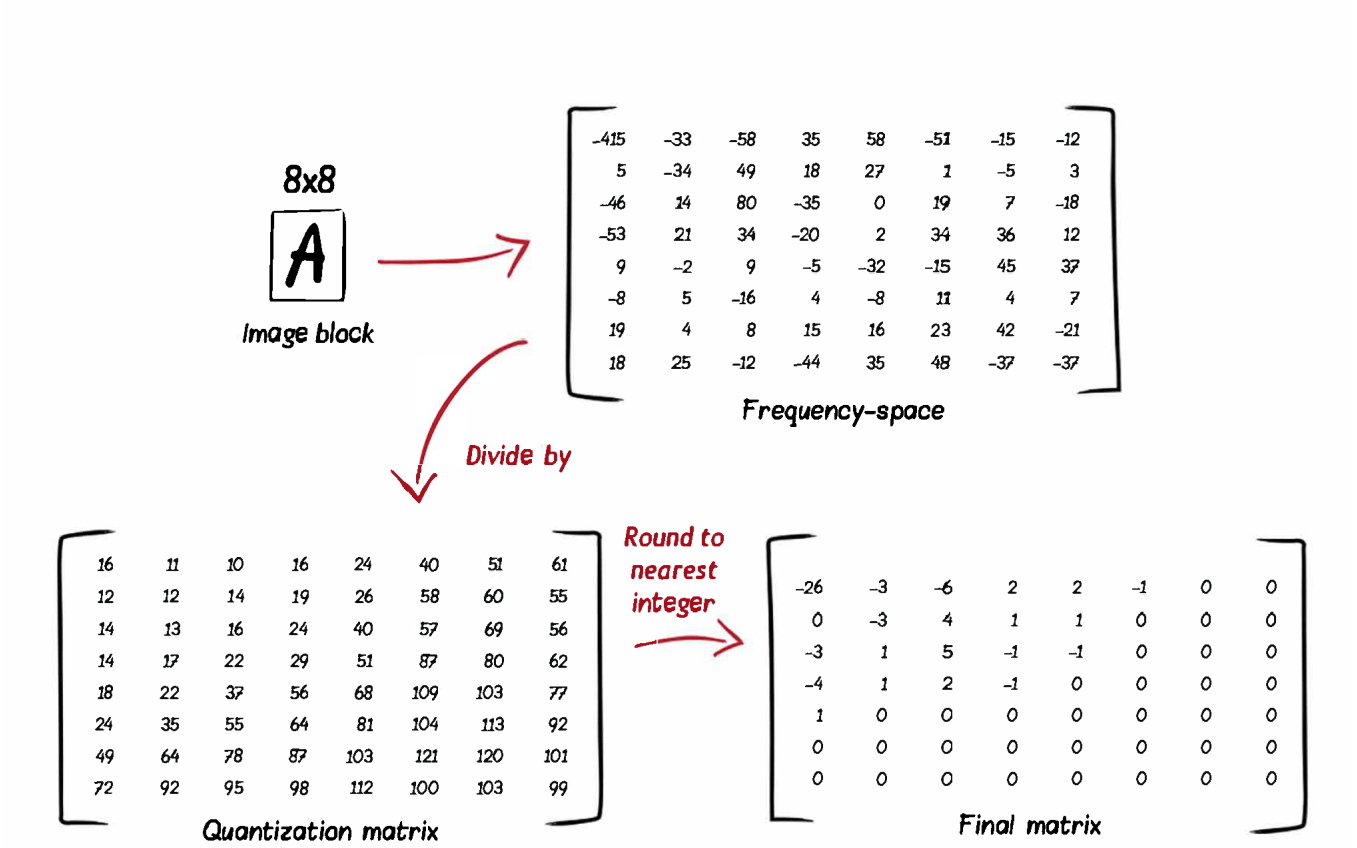
- Each value is the strength of each frequency component, starting from a steady level (DC–average of all pixels) and each step is an increase in frequency in the x and y direction in the 8×8 block

JPEG: Quantization of frequency coefficients

JPEG steps:

1. Separate luminance from color $\longrightarrow$ JPEG first converts RGB images to the YCbCr color space
2. Apply downsampling to the chroma components, i.e., reduce its resolution
3. Each 8×8 block is converted to a frequency-domain representation by means of a discrete cosine transform (DCT) $\longrightarrow$ 8×8 block of frequency coefficient values
4. Quantize the frequency coefficients, with different quantization for the luma and chroma components!

JPEG: Quantization of frequency coefficients



- Values of the quantization matrix chosen to preserve low-frequency information and discard high-frequency detail $\longrightarrow$ reduce most of the (less important) high frequency coefficients to zero
- Quantization matrix different for luma and chroma components

JPEG: Compression with run-length and Huffman coding

1. Separate luminance from color $\longrightarrow$ JPEG first converts RGB images to the YCbCr color space
2. Apply downsampling to the chroma components, i.e., reduce its resolution
3. Each 8×8 block is converted to a frequency-domain representation by means of a discrete cosine transform (DCT) $\longrightarrow$ 8×8 block of frequency coefficient values
4. Quantize the frequency coefficients, with different quantization for the luma and chroma components!
5. Read the quantized matrices in zigzag $\longrightarrow$ one-dimensional vector with many consecutive zeroes
6. Apply run-length coding
7. Apply Huffman coding to the resulting sequence of symbols

JPEG: Run-length and Huffman coding

After Quantization

-24	-23	0	0	0	0	0	0
-19	4	1	0	0	0	0	0
5	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	0	0	0	0	0	0
0	0	0	0	0	0	0	0

- Read the matrix in zigzag:

$$-24, -23, -19, 5, 4, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, \ldots, 0$$

- Apply run-length coding:

$$(0, -24), (0, -23), (0, -19), (0, 5), (0, 4), (2, 1), (4, 1), (0, 0)$$

- Apply Huffman coding to the resulting sequence

JPEG: Example

raw compression 4:1