T1 – The image: JPEG, JPEG 2000 & other ones

Universitat Pompeu Fabra Barcelona

JPEG

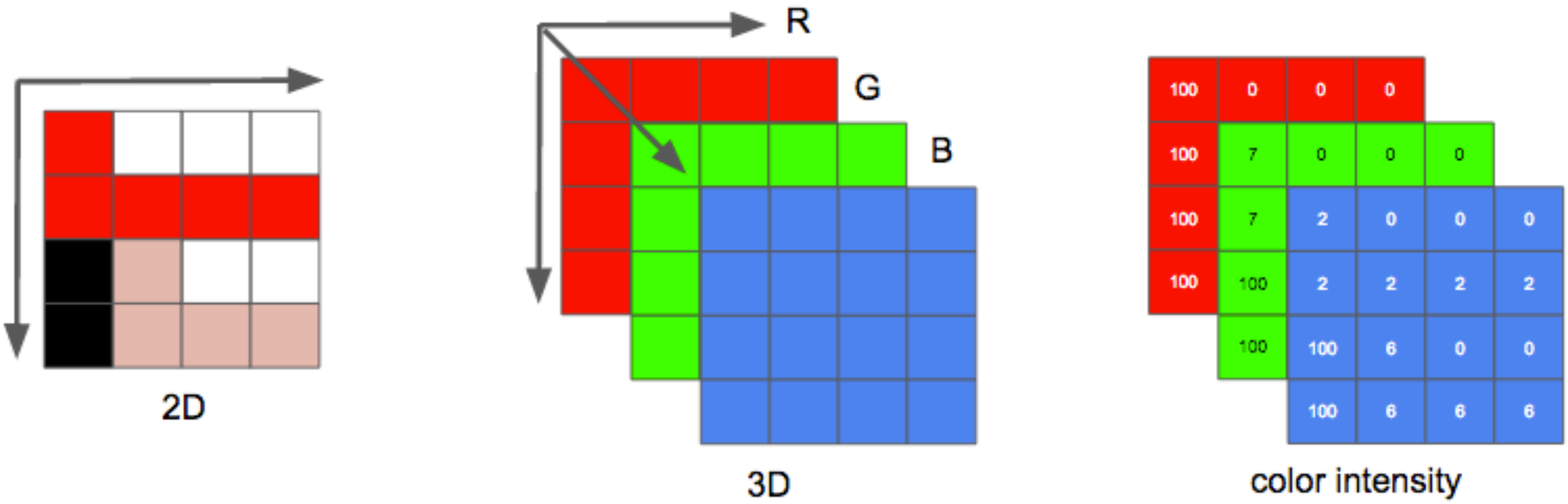
Pixel (picture element)

Represents the intensity (usually a numeric value) of a given color.

For example:

Red Pixel: 0 of green + 0 of blue + maximum of red

Pink Pixel: 192 of green + 203 of blue + 255 of red


2D
R
G
B
3D
100
0
0
0
100
7
0
0
0
100
7
2
0
0
0
100
100
2
2
2
2
100
100
6
0
0
100
6
6
6
color intensity

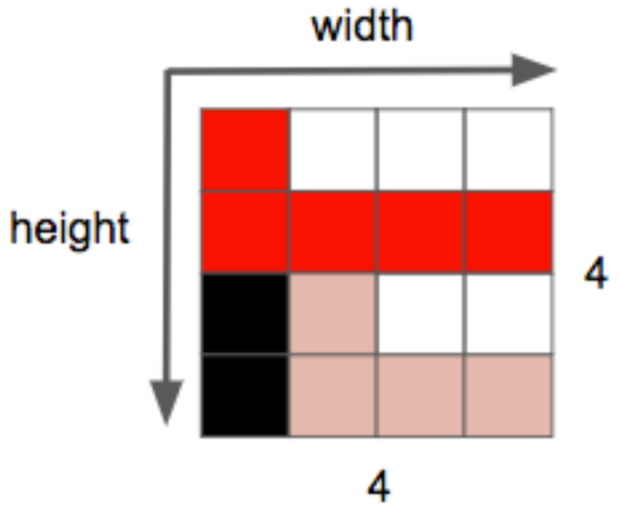
Other ways to encode a color image

Many other possible models may be used to represent the colors that make up an image. We could, for instance, use an indexed palette where we'd only need a single byte to represent each pixel instead of the 3 needed when using the RGB model. In such a model we could use a 2D matrix instead of a 3D matrix to represent our color, this would save on memory but yield fewer color options.

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F

Universitat Pompeu Fabra Barcelona

Resolution: number of pixels in 1 dimension



Joint
Photographic
Experts
Group

Universitat Pompeu Fabra Barcelona

ISO norm created in 1983

Lossy compression method

It also stands for files (.jpg)

Supports true color (24 bits):

That means 16.777.216 different colors

Universitat Pompeu Fabra Barcelona

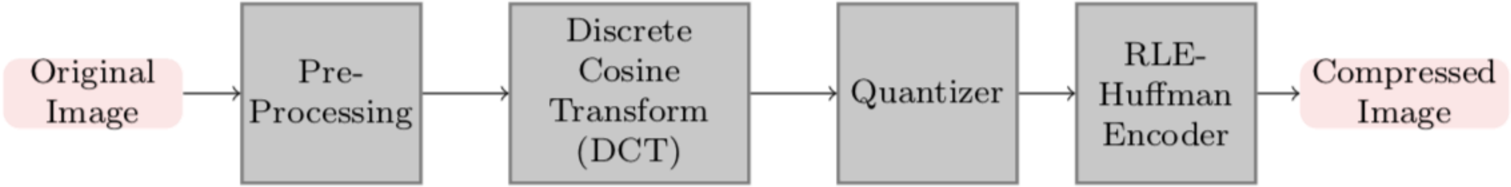
It's compression is a mathematical model based. This is the formula

$$G_{u,v} = \alpha(u)\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7} g_{x,y}\cos\left[\frac{\pi}{8}\left(x+\frac{1}{2}\right)u\right]\cos\left[\frac{\pi}{8}\left(y+\frac{1}{2}\right)v\right]$$

$$\alpha_p(n) = \begin{cases} \sqrt{\frac{1}{8}}, & \text{if } n = 0 \\ \sqrt{\frac{2}{8}}, & \text{otherwise} \end{cases}$$

Universitat Pompeu Fabra Barcelona

...and this is the encoding diagram



Universitat Pompeu Fabra Barcelona

...and of course, the decoding diagram

Universitat Pompeu Fabra Barcelona

Our human eye:

- We recognize objects equally well regardless of image size
- Recognition speed doesn't depend on image size

PROS: nice compression of data. Extremely used in computing and Internet

Universitat Pompeu Fabra Barcelona

CONS: Lossy method

Universitat Pompeu Fabra Barcelona

a. Original image

b. With 10:1 compression

FIGURE 27-15
Example of JPEG distortion. Figure (a) shows the original image, while (b) and (c) shows restored images using compression ratios of 10:1 and 45:1, respectively. The high compression ratio used in (c) results in each 8×8 pixel group being represented by less than 12 bits.

c. With 45:1 compression

DCT – Discrete Cosine Transform.

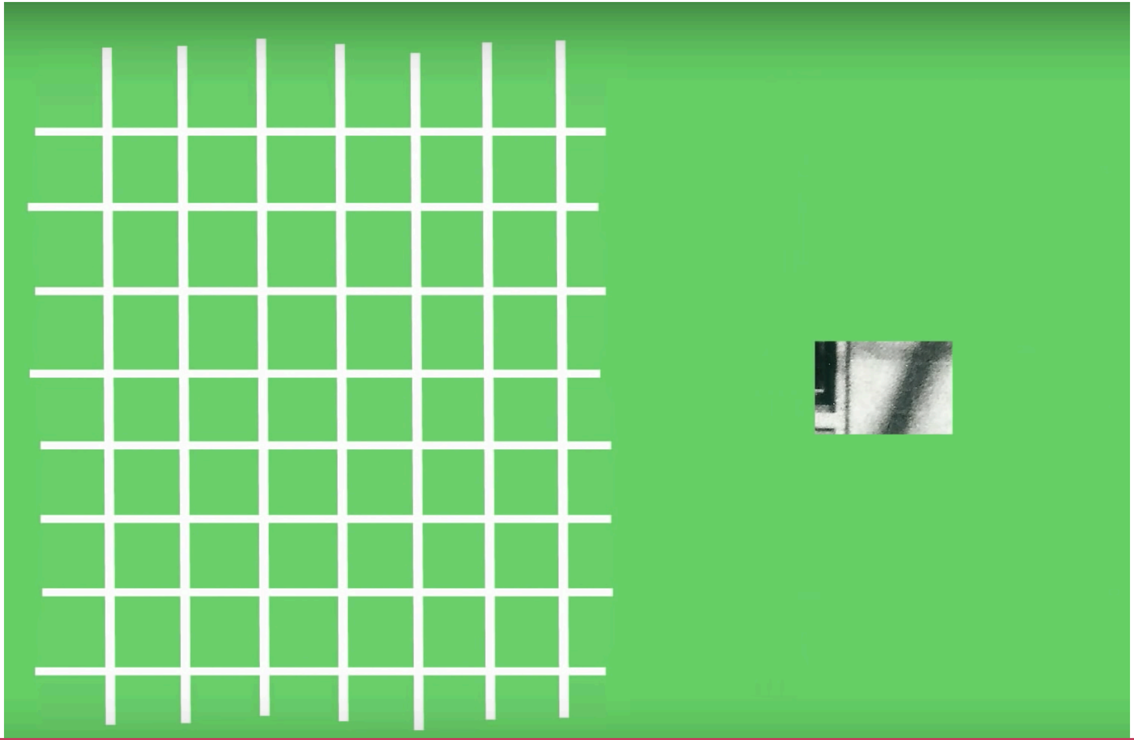
How does it work?

Universitat Pompeu Fabra Barcelona

RAW Data: analogic B&W picture

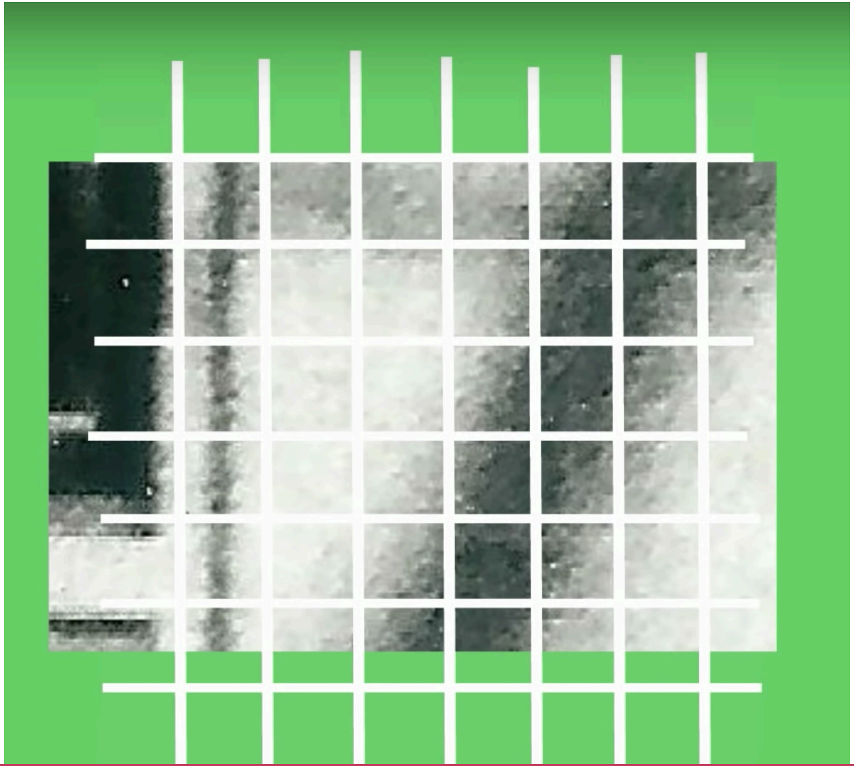
Universitat Pompeu Fabra Barcelona

We take a small part into the grid

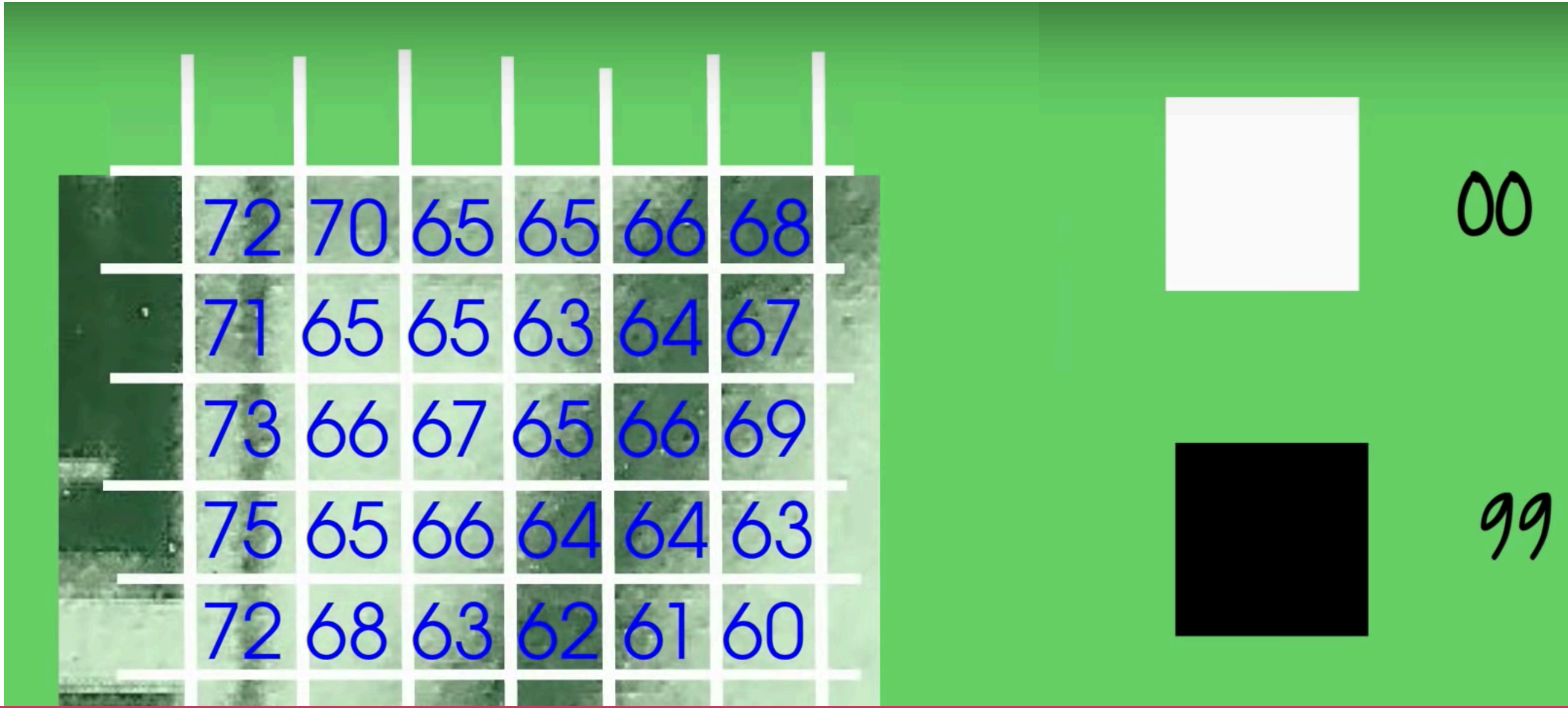


Universitat Pompeu Fabra Barcelona

We take a small part into the grid

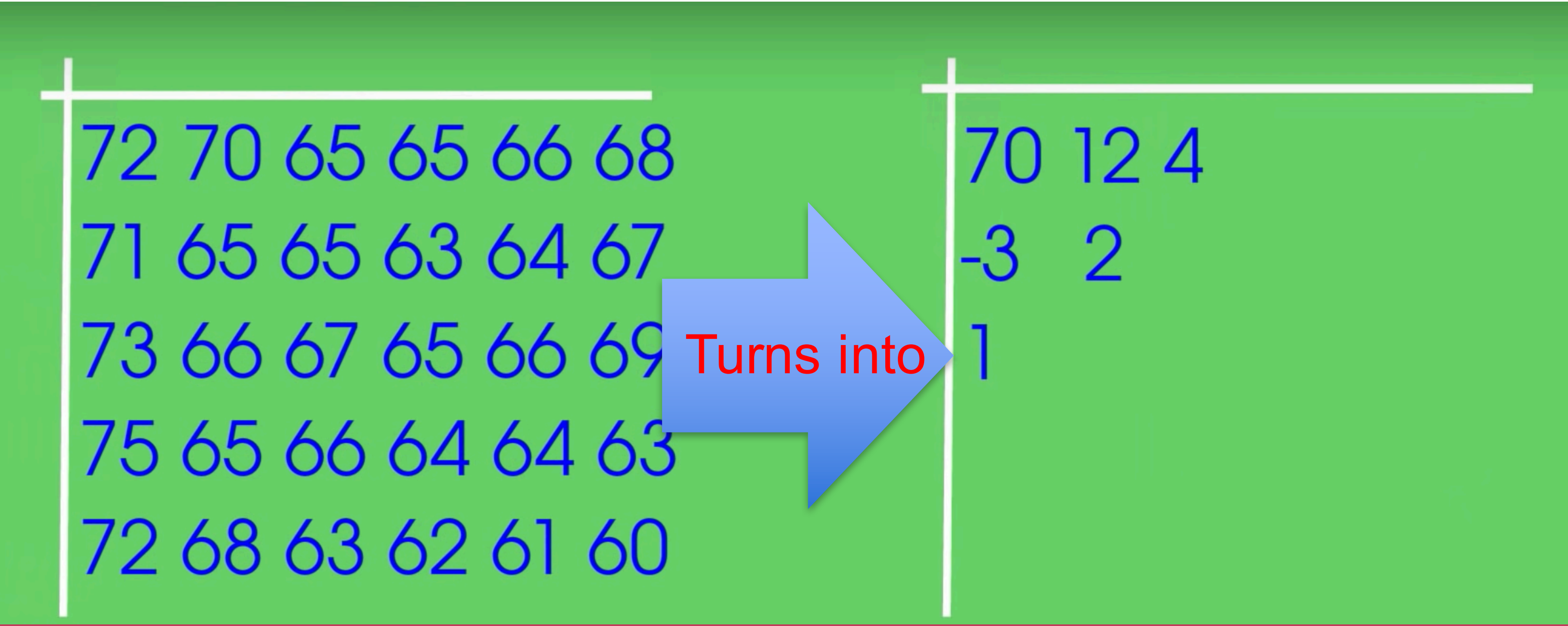


We quantize

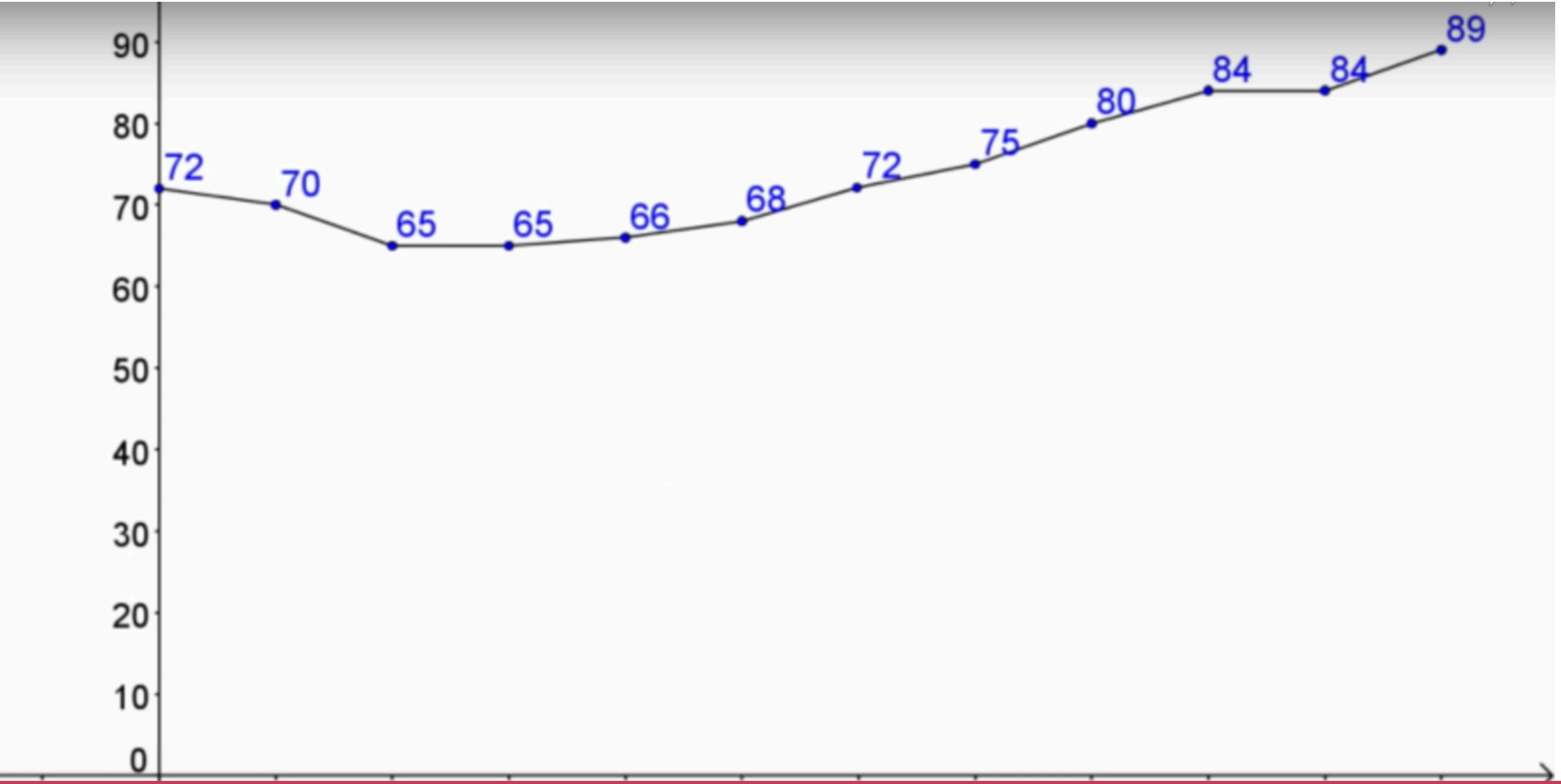

72 70 65 65 66 68
71 65 65 63 64 67
73 66 67 65 66 69
75 65 66 64 64 63
72 68 63 62 61 60
00
99

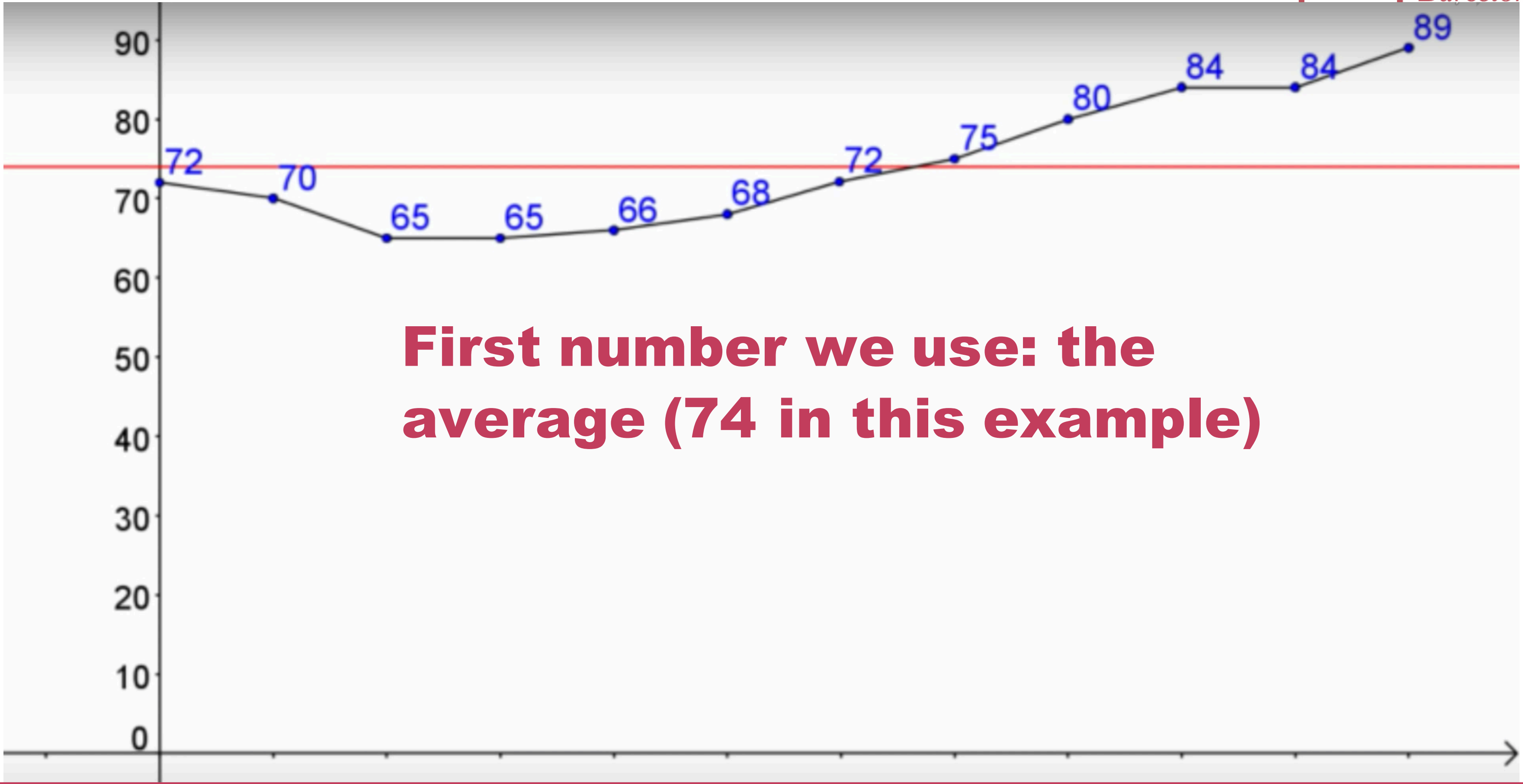
Universitat Pompeu Fabra Barcelona

How do we compress?

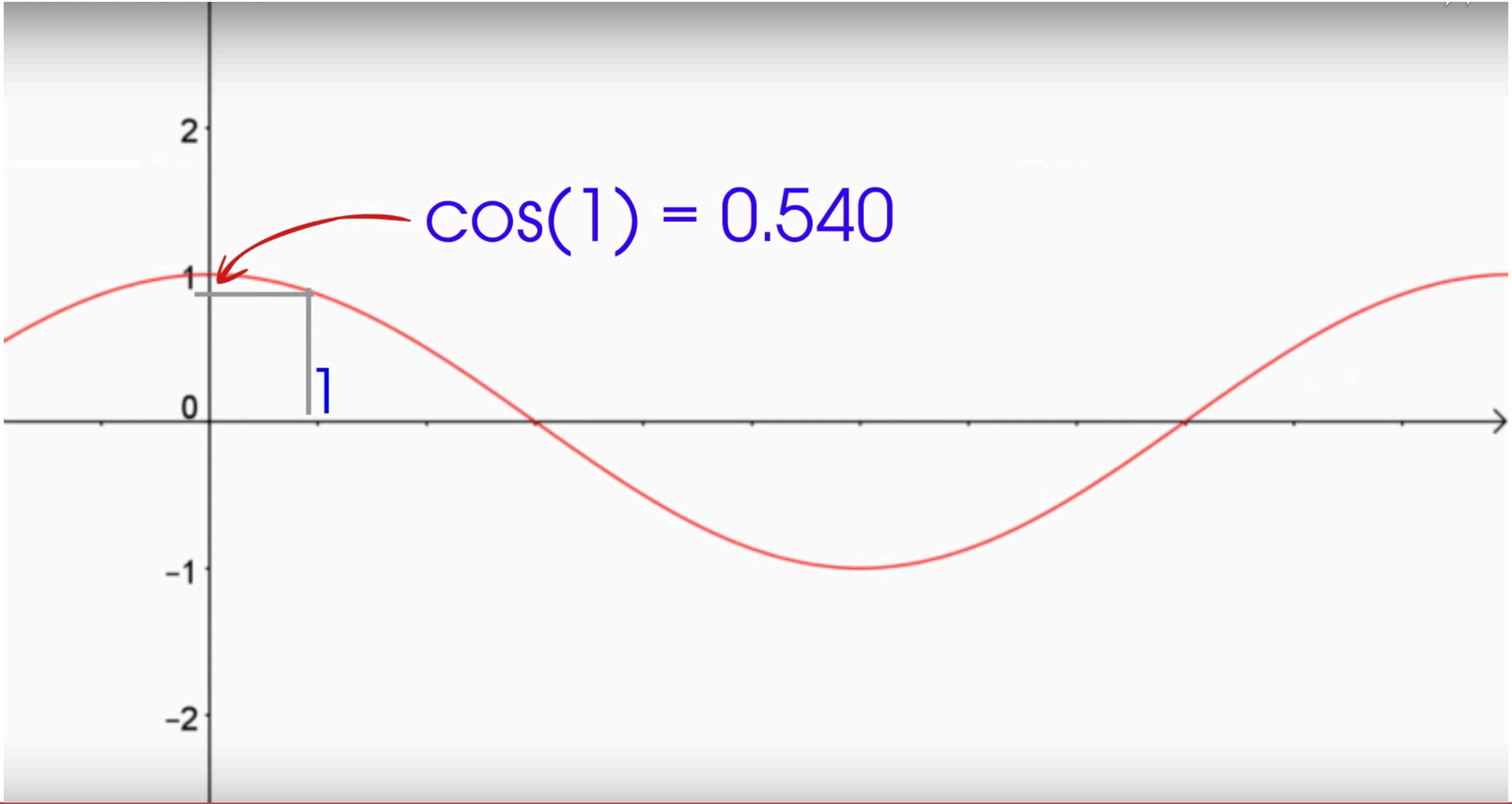


72 70 65 65 66 68 72 75 80 84 84 89

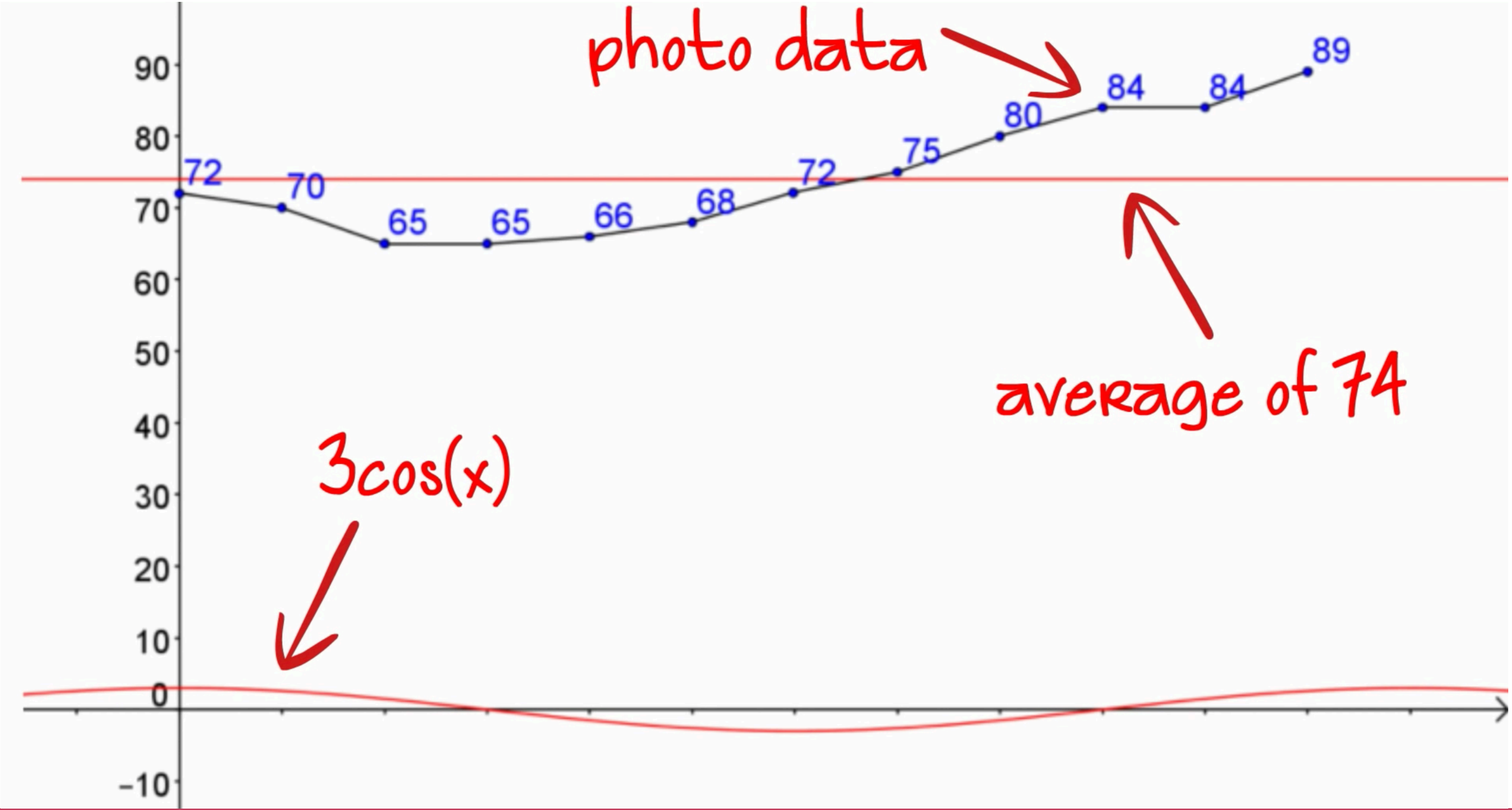

90
80
70
60
50
40
30
20
10
0
72
70
65
65
66
68
72
75
80
84
84
89

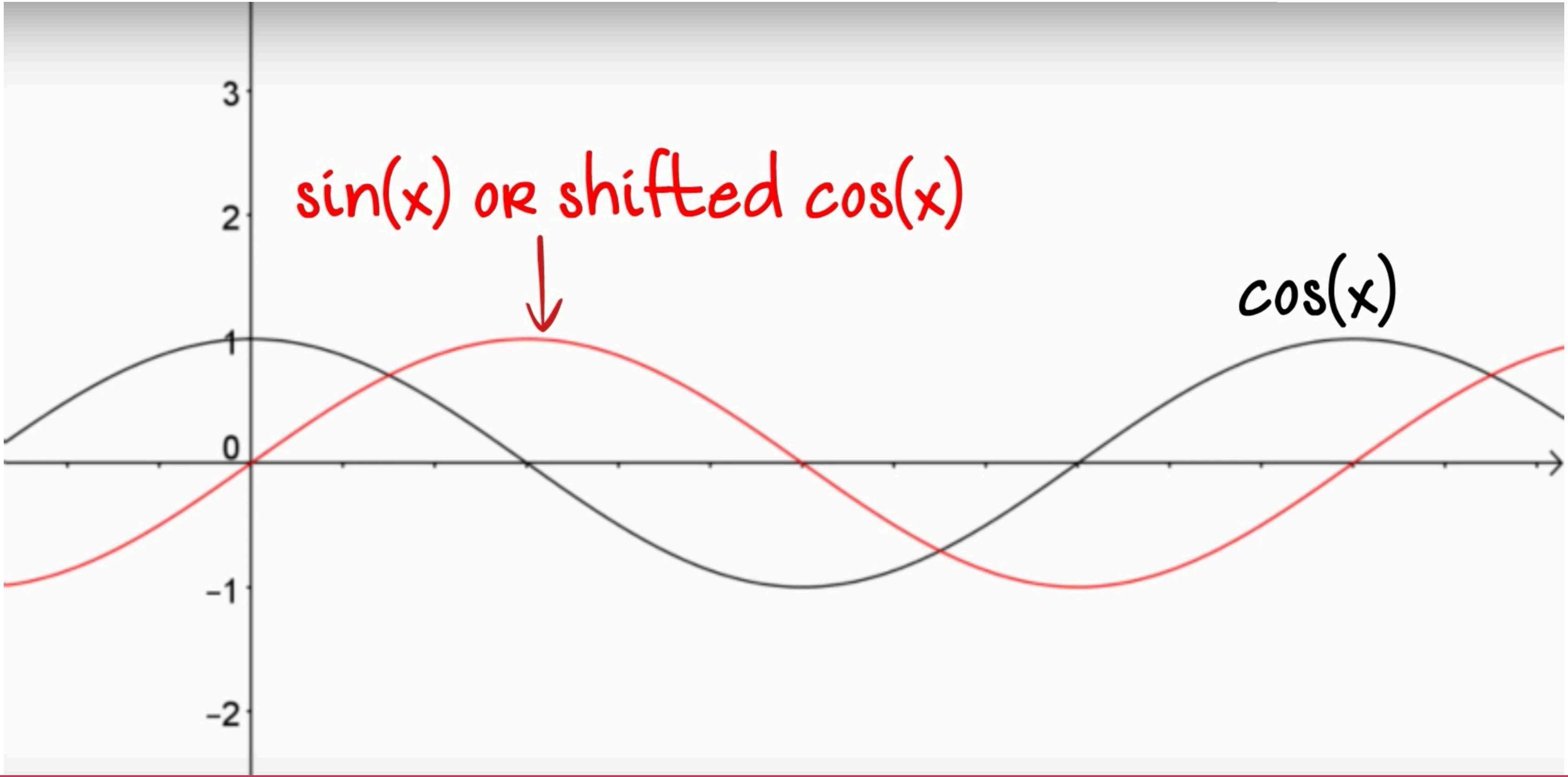


First number we use: the average (74 in this example)

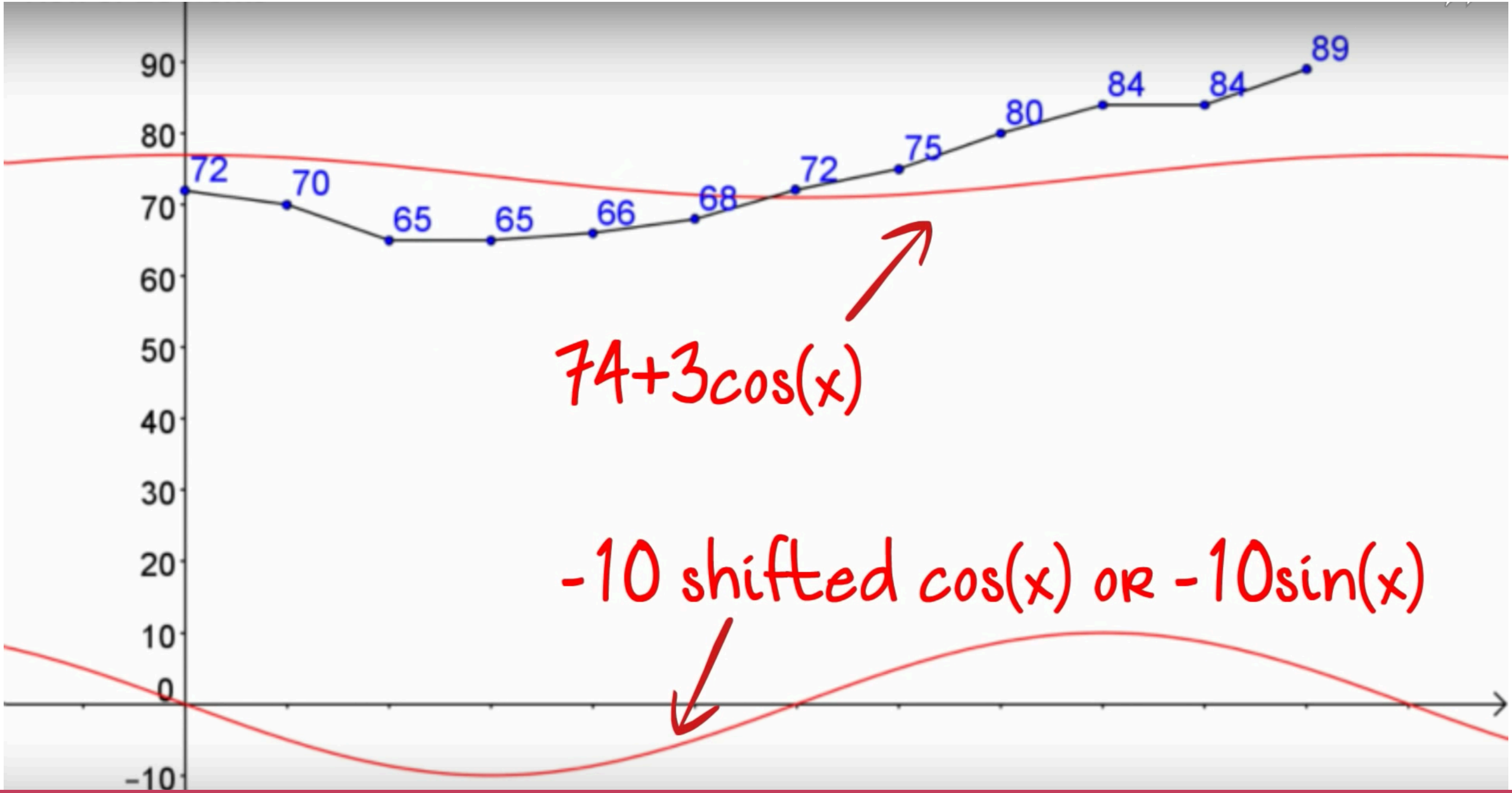

cos(1) = 0.540
2
1
0
1
-1
-2

Universitat Pompeu Fabra Barcelona




sin(x) or shifted cos(x)
cos(x)
3
2
1
0
-1
-2

Universitat Pompeu Fabra Barcelona



Universitat Pompeu Fabra Barcelona

Universitat Pompeu Fabra Barcelona

Remember the formula?

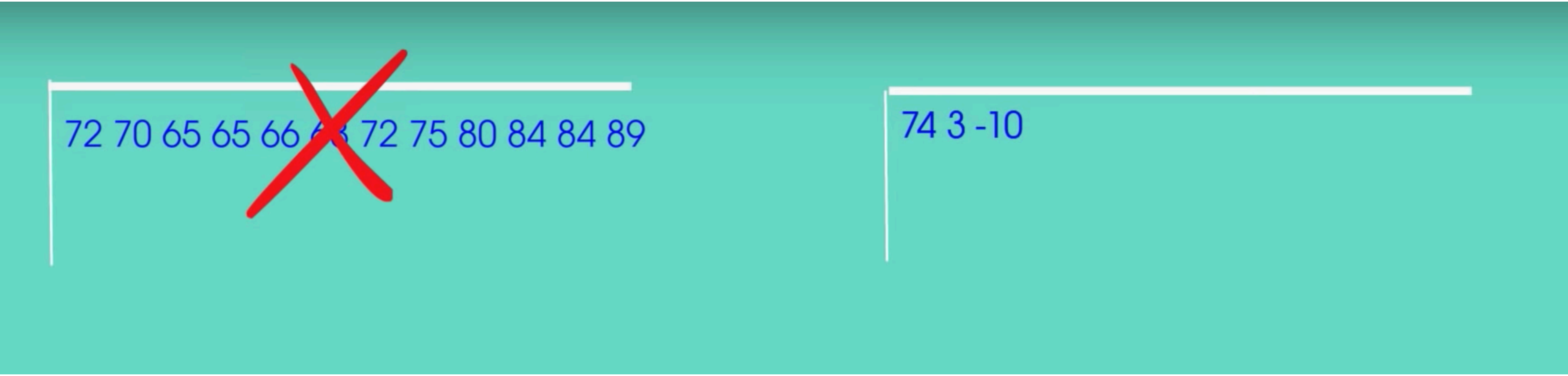
$$G_{u,v} = \alpha(u)\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7} g_{x,y}\cos\left[\frac{\pi}{8}\left(x+\frac{1}{2}\right)u\right]\cos\left[\frac{\pi}{8}\left(y+\frac{1}{2}\right)v\right]$$

$$\alpha_p(n) = \begin{cases} \sqrt{\frac{1}{8}}, & \text{if } n = 0 \\ \sqrt{\frac{2}{8}}, & \text{otherwise} \end{cases}$$

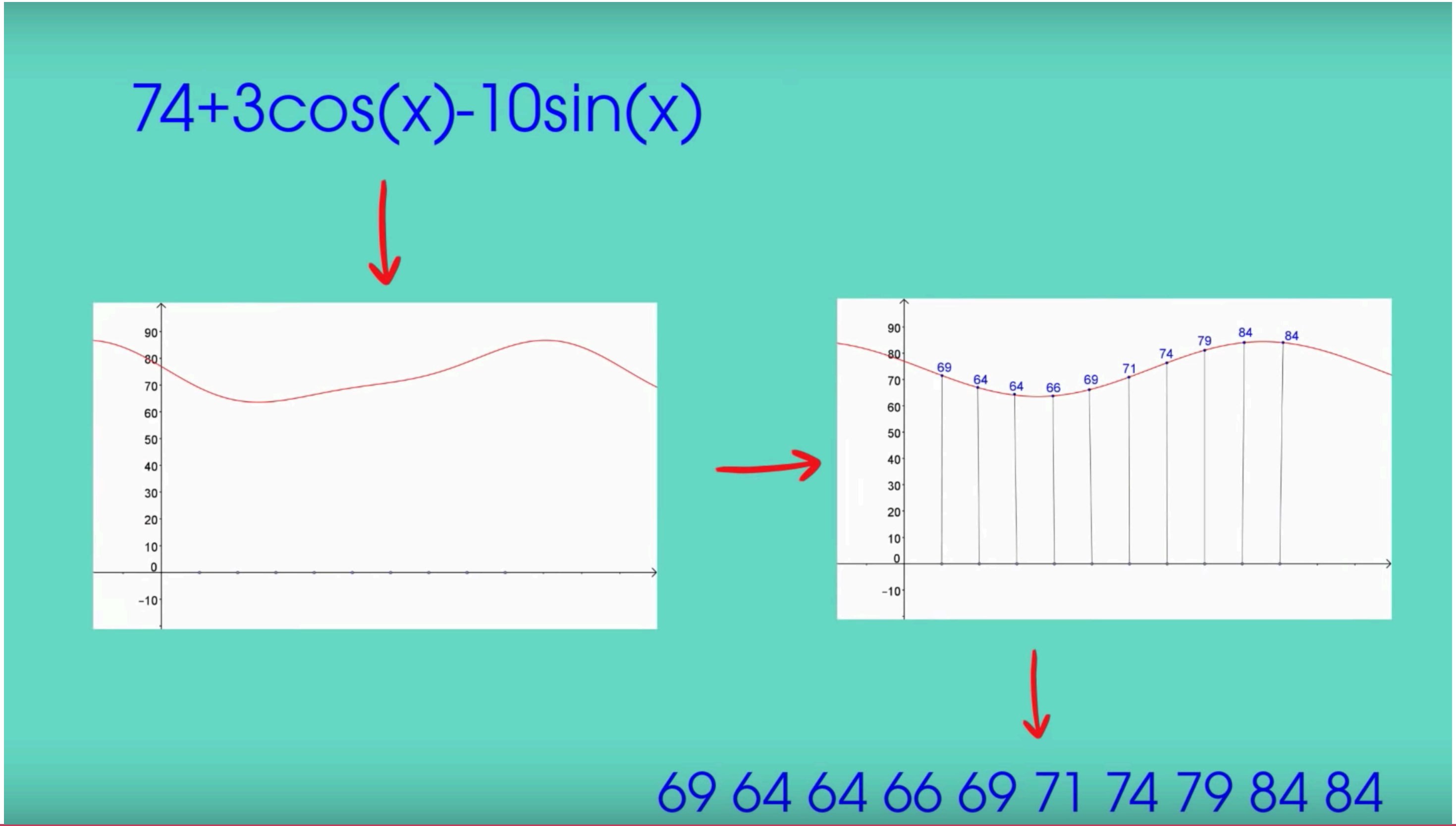
Remember the formula?

$$G_{u,v} = \alpha(u)\alpha(v) \sum_{x=0}^{7} \sum_{y=0}^{7} g_{x,y} \cos\left[\frac{\pi}{8}\left(x+\frac{1}{2}\right)u\right] \cos\left[\frac{\pi}{8}\left(y+\frac{1}{2}\right)v\right]$$

$$\alpha_p(n) = \begin{cases} \sqrt{\frac{1}{8}}, & \text{if } n = 0 \\ \sqrt{\frac{2}{8}}, & \text{otherwise} \end{cases}$$


72 70 65 65 66 72 75 80 84 84 89
74 3 -10

Universitat Pompeu Fabra Barcelona



Universitat Pompeu Fabra Barcelona

72 70 65 65 66 68 72 75 80 84 84 89

or

74 3 -10

75% less storage space!

4 times faster!

Universitat Pompeu Fabra Barcelona

Remember run-length encoding? It's also applied here

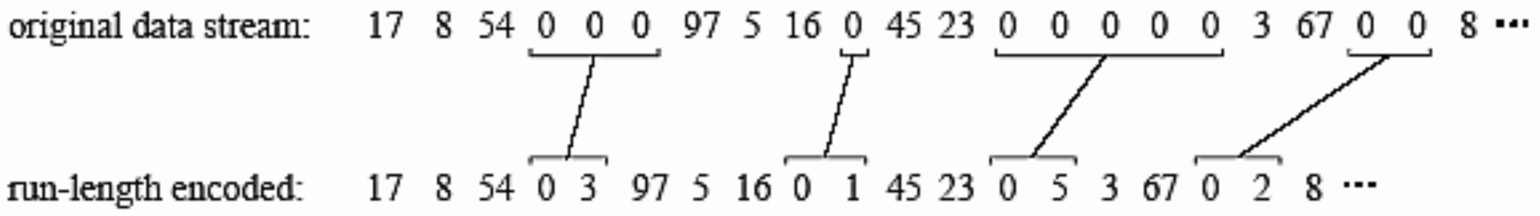
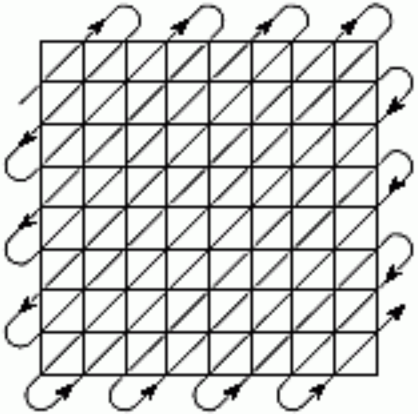
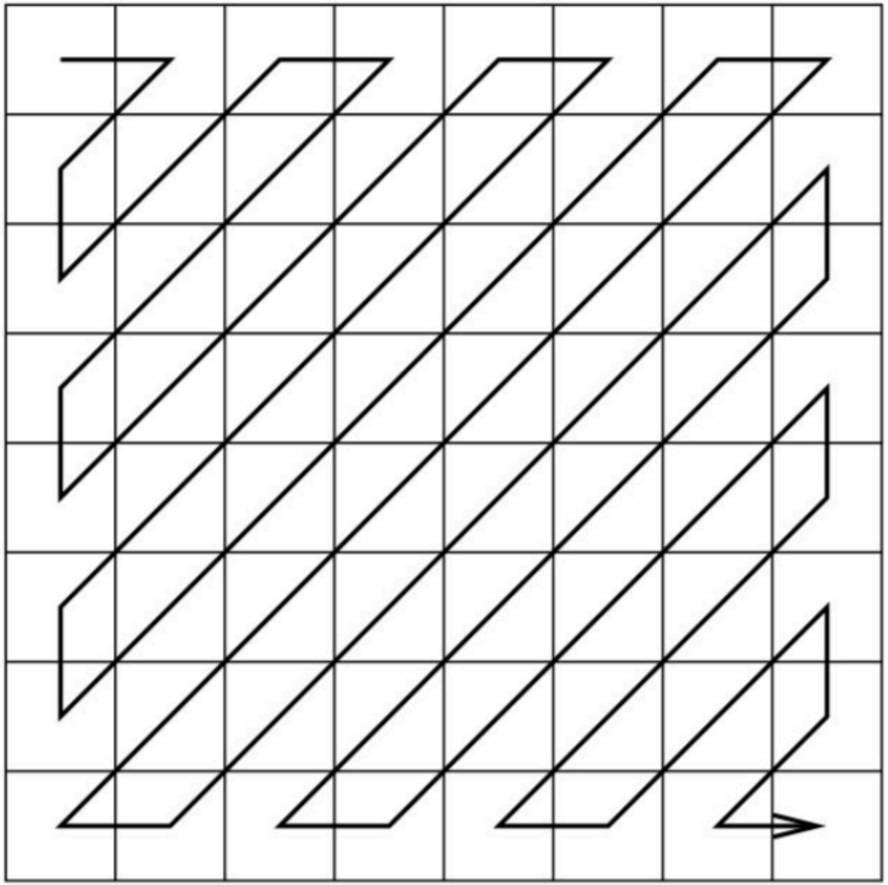


FIGURE 27-1
Example of run-length encoding. Each run of zeros is replaced by two characters in the compressed file: a zero to indicate that compression is occurring, followed by the number of zeros in the run.

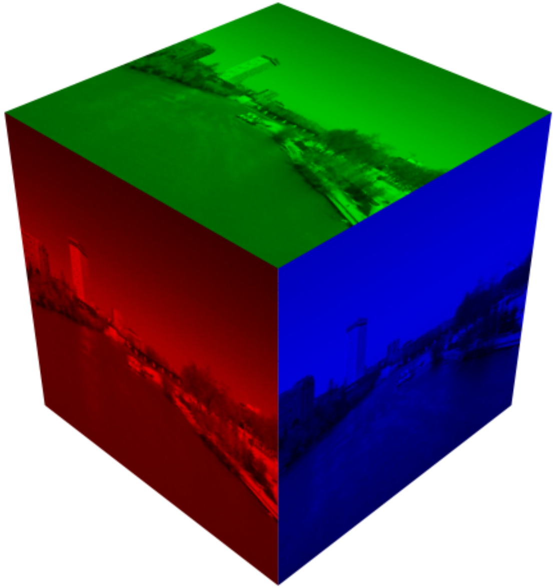
FIGURE 27-14
JPEG serial conversion. A serpentine pattern used to convert the 8×8 DCT spectrum into a linear sequence of 64 values. This places all of the high frequency components together, where the large number of zeros can be efficiently compressed with run-length encoding.





What about color?

It’s doing x3 the opperation

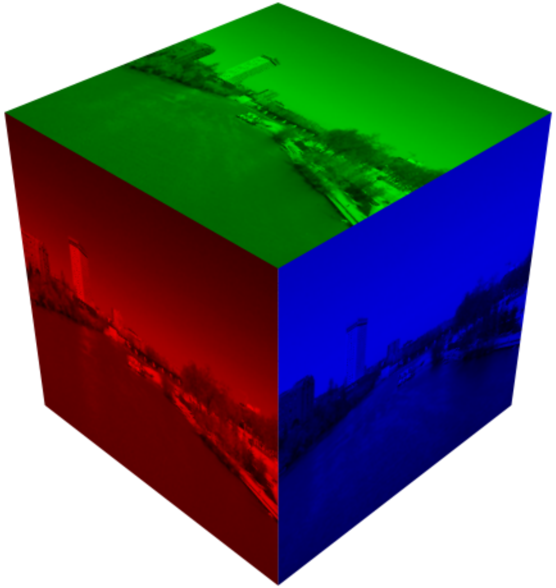


Universitat Pompeu Fabra Barcelona

Used to be

8 bit series for RGB

It's like compressing 3 different and sumarizing them…



But JPEG introduced YCbCr model

Y: Luminance

Cb: Chroma blue difference

Cr: Chroma red difference

Universitat Pompeu Fabra Barcelona

But JPEG introduced YCbCr model

$$Y' = K_R \cdot R' + K_G \cdot G' + K_B \cdot B'$$

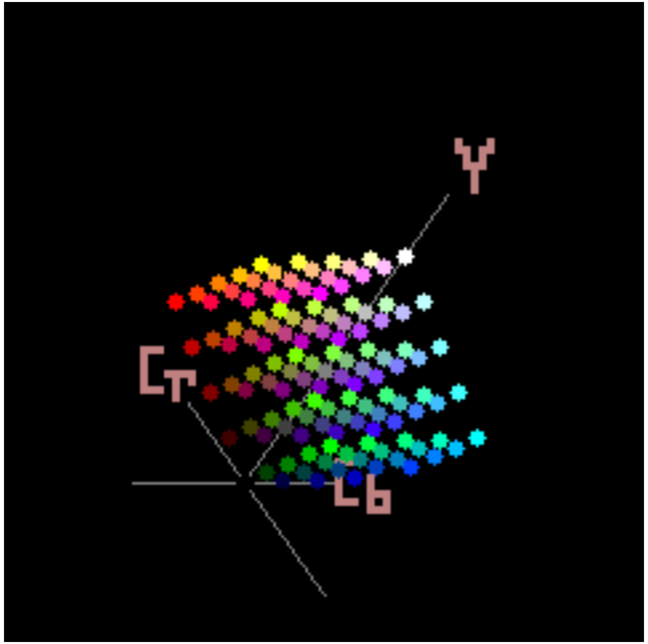
$$P_B = \frac{1}{2} \cdot \frac{B' - Y'}{1 - K_B}$$

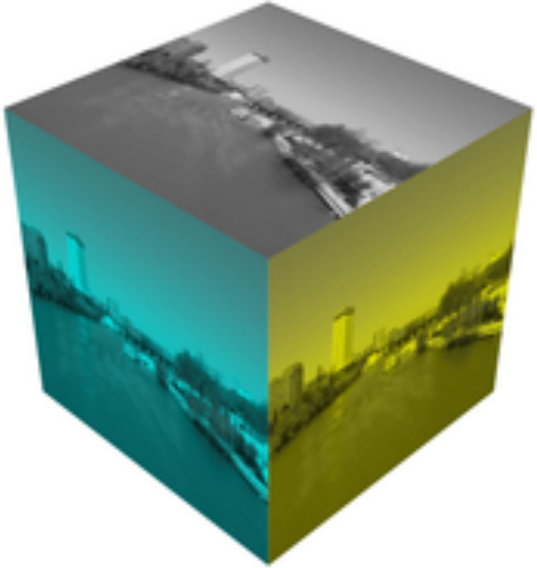
$$P_R = \frac{1}{2} \cdot \frac{R' - Y'}{1 - K_R}$$

YCbCr formula from RGB

```
Y  =       0,257 * R + 0,504 * G + 0,098 * B + 16
Cb = U = -0,148 * R - 0,291 * G + 0,439 * B + 128
Cr = V =  0,439 * R - 0,368 * G - 0,071 * B + 128
```

```
B = 1,164 * (Y - 16)                      + 2,018 * (U - 128)
G = 1,164 * (Y - 16) - 0,813 * (V - 128) - 0,391 * (U - 128)
R = 1,164 * (Y - 16) + 1,596 * (V - 128)
```


Y
Cr
Cb



PLEASE NOTICE: don't call it YPbPr, which is for analog TV!!! (but based on same concept)

How to solve the quality loss problem?

The JPEG 2000 solution

Joint Photographic Experts Group

Universitat Pompeu Fabra Barcelona

JPEG2000:

- New standard presented year 2000 in order to substitute JPEG

- Files extension is .jp2

- Technologically is probably the best possible engineering solution to the problem they had (and that's why we're studying it!)

upf. Universitat Pompeu Fabra Barcelona

JPEG2000:

·However it failed to standarize, as it required the hardware to adapt to this codec (new technology and backwards incompatible with JPEG)

Universitat Pompeu Fabra Barcelona

JPEG2000:

- It focuses on the big changes of data

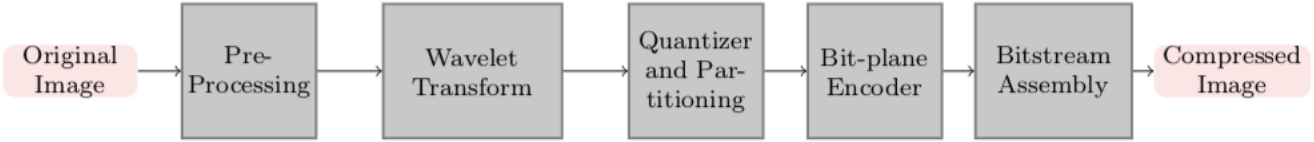
Universitat Pompeu Fabra Barcelona

Remember this slide?

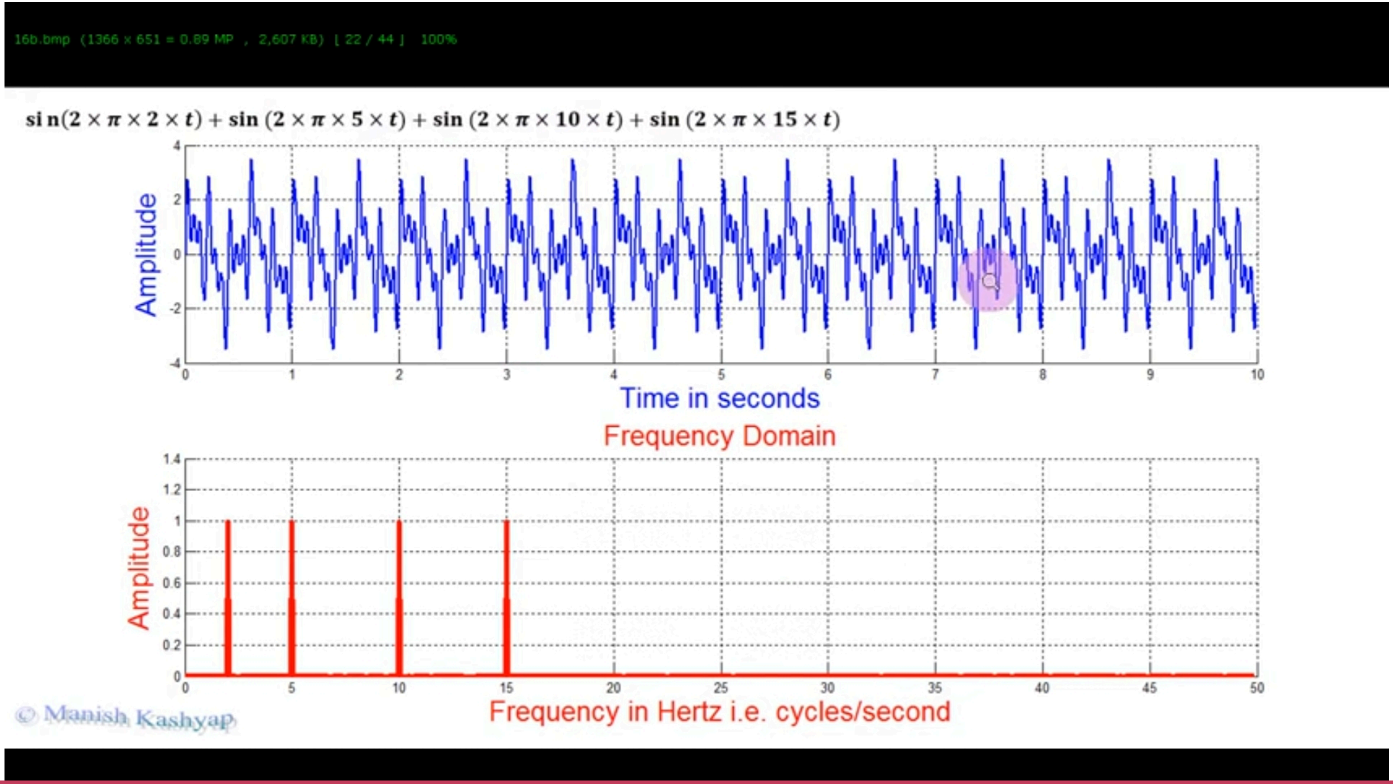
Universitat Pompeu Fabra Barcelona

Figure 3.9: Recovered images after JPEG compression with ratios $k = 1, 5, 10, 20$.

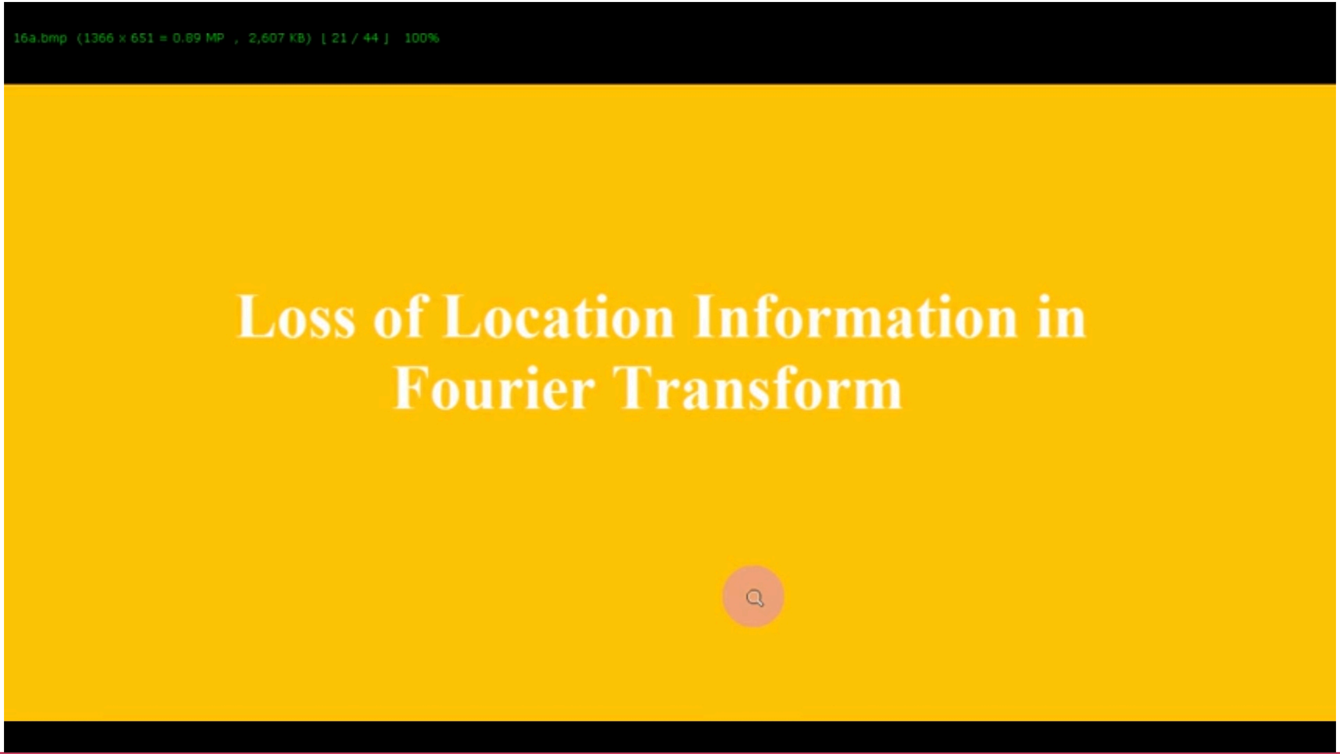
Diagram of the JPEG2000 encoder

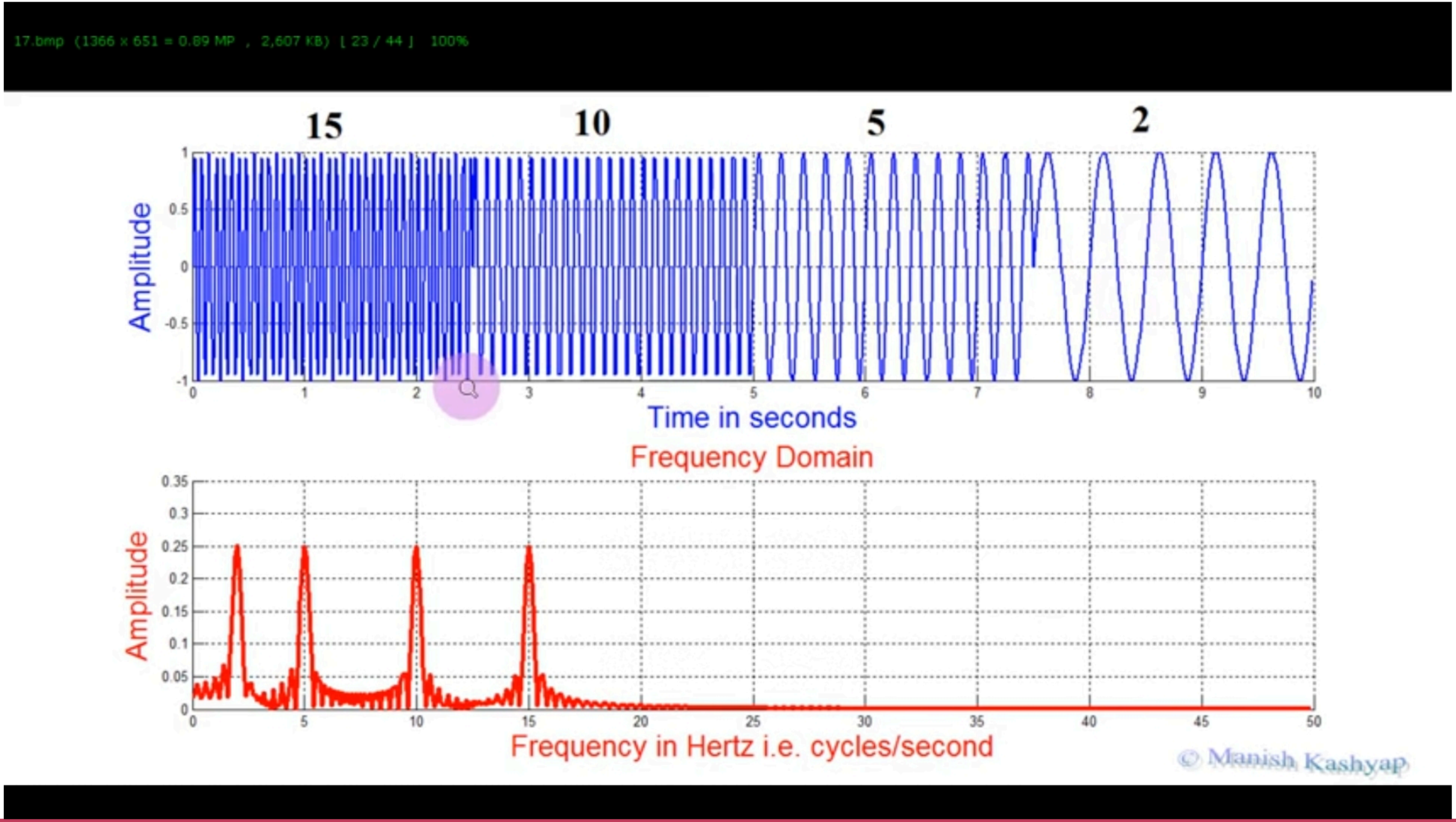


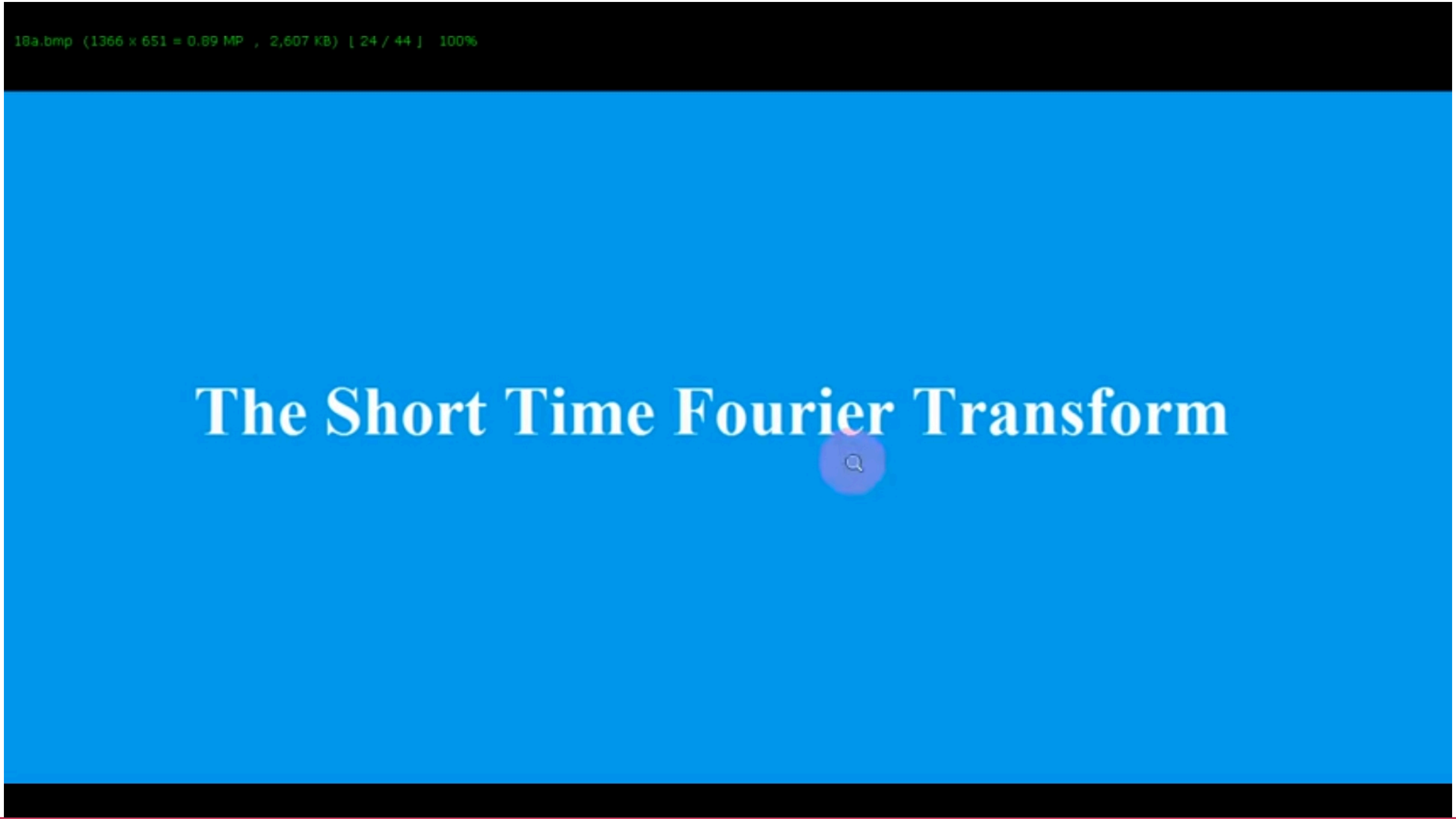
Let’s recap. How to solve the quality loss problem? Where was the loss happening?



16b.bmp (1366 x 651 = 0.89 MP , 2,607 KB) [22 / 44] 100%
sin(2 × π × 2 × t) + sin (2 × π × 5 × t) + sin (2 × π × 10 × t) + sin (2 × π × 15 × t)
Amplitude
Time in seconds
Frequency Domain
Amplitude
Frequency in Hertz i.e. cycles/second
© Manish Kashyap


16a.bmp (1366 x 651 = 0.89 MP , 2,607 KB) [21 / 44] 100%
Loss of Location Information in
Fourier Transform

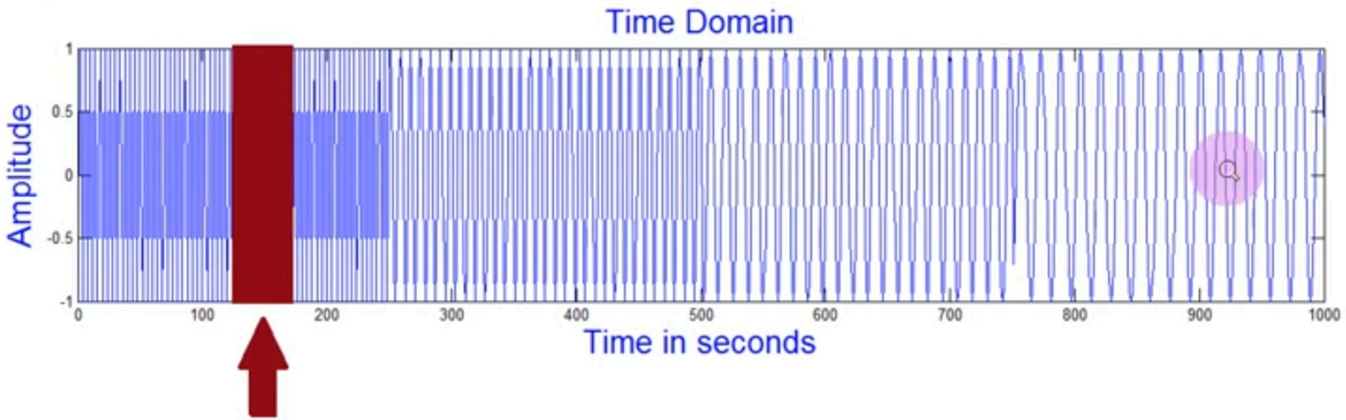

17.bmp (1366 x 651 = 0.89 MP , 2,607 KB) [23 / 44] 100%
15
10
5
2
Amplitude
Time in seconds
Frequency Domain
Amplitude
Frequency in Hertz i.e. cycles/second
© Manish Kashyap


18a.bmp (1366 x 651 = 0.89 MP , 2,607 KB) [24 / 44] 100%
The Short Time Fourier Transform

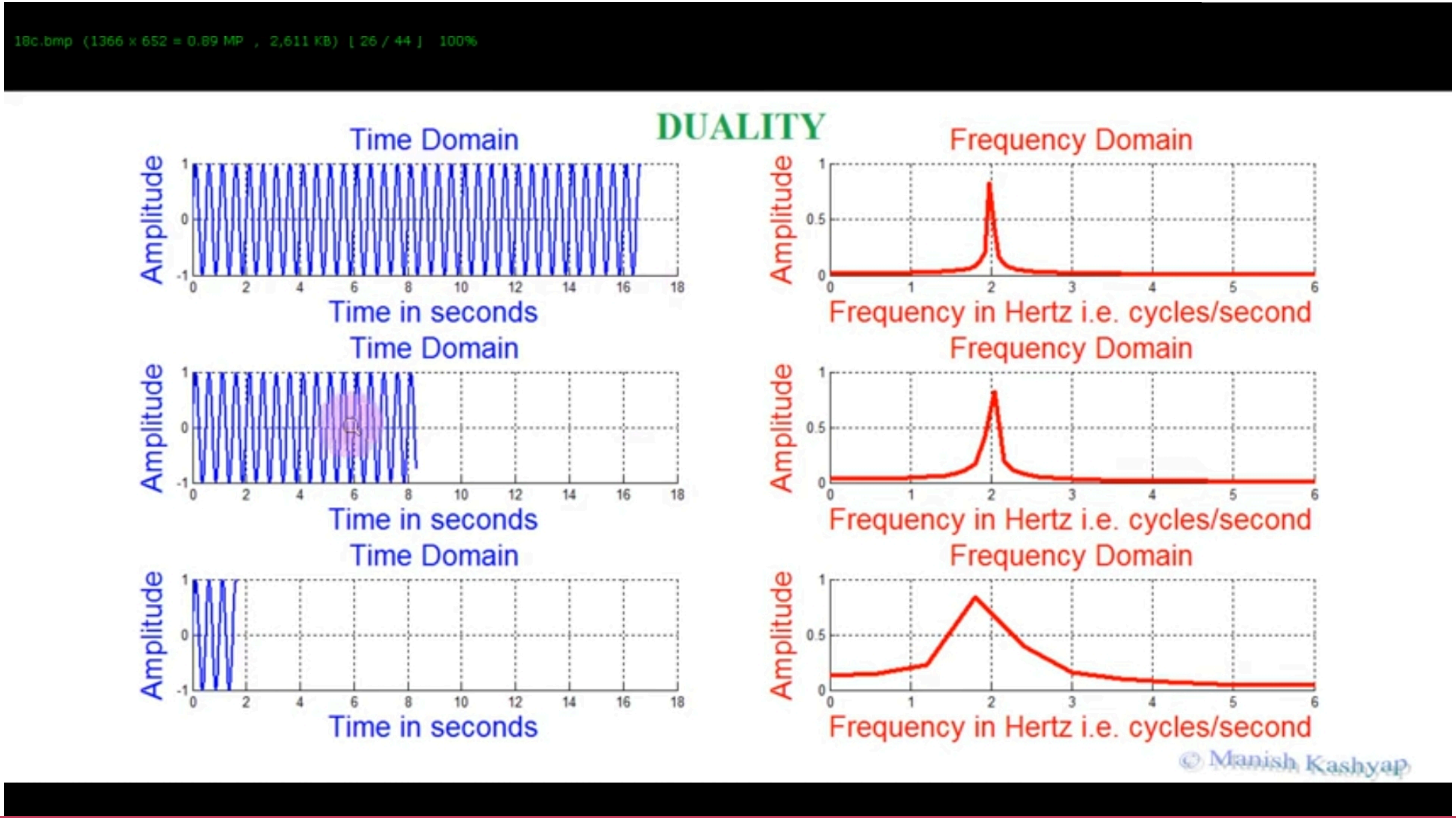
Universitat Pompeu Fabra Barcelona

18b.bmp (1366 x 651 = 0.89 MP , 2,607 KB) [25 / 44] 100%

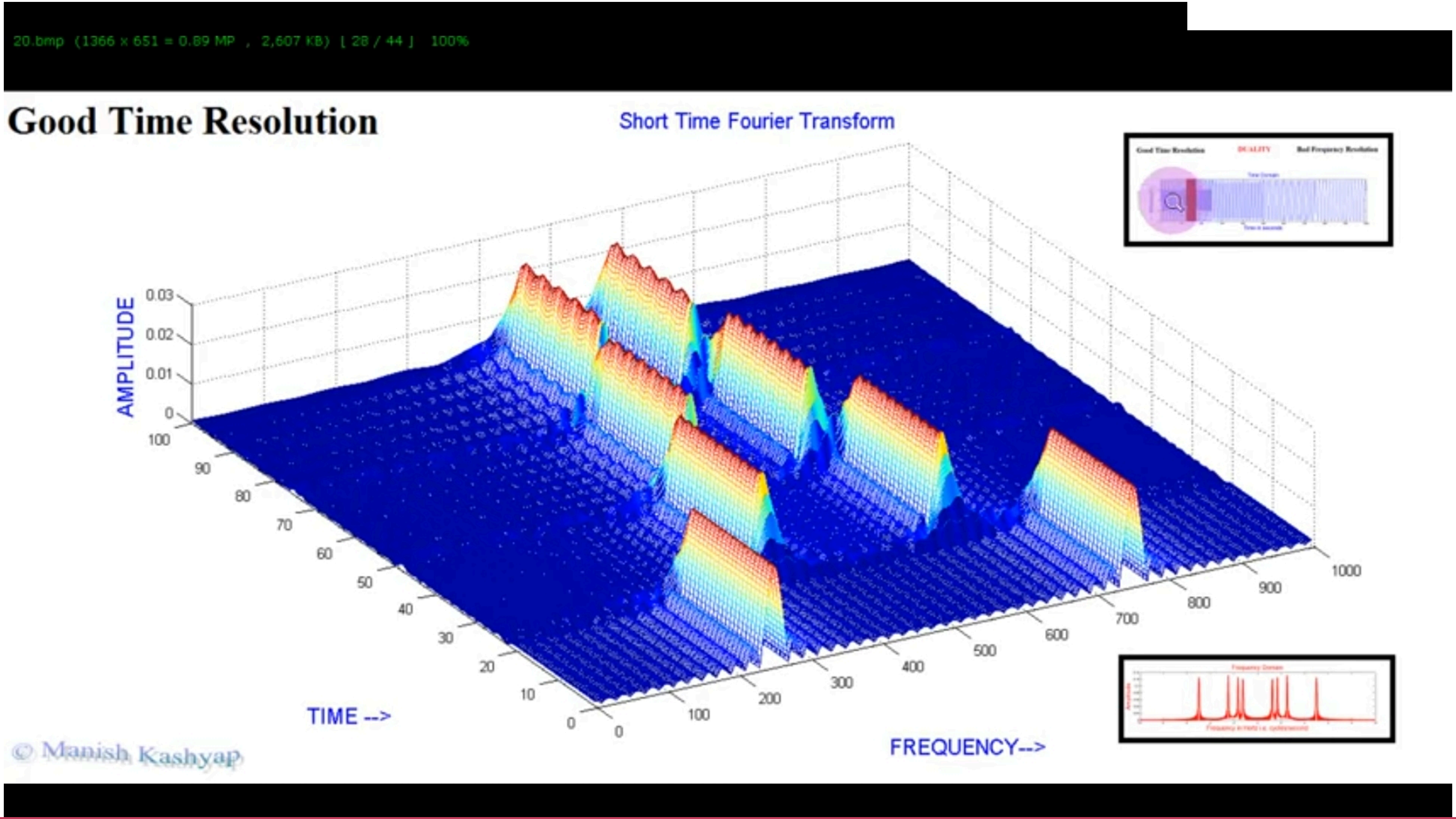
Good Time Resolution **DUALITY** **Bad Frequency Resolution**



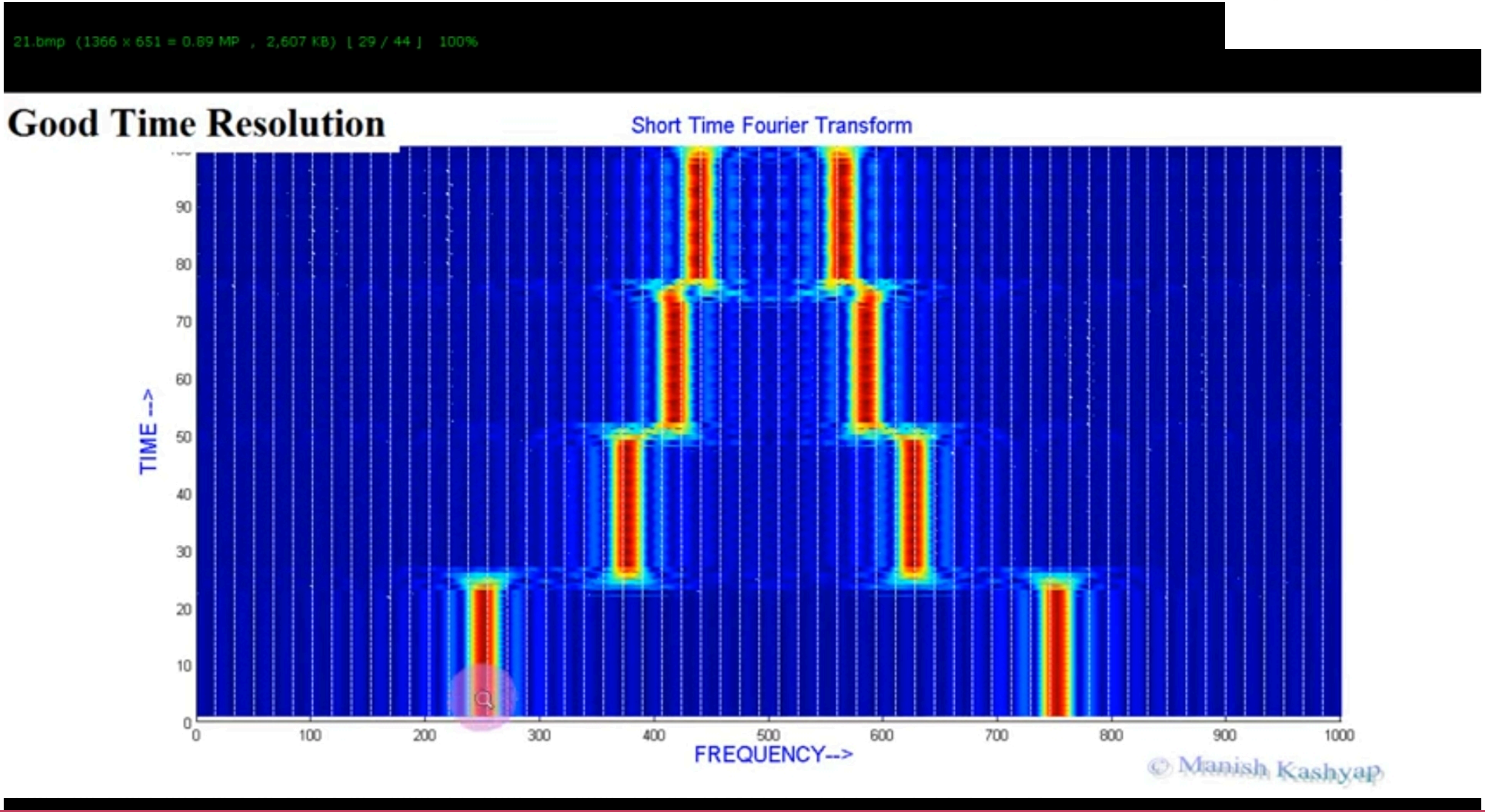
Start from the left, take Fourier Transform of the portion of signal overlaping with window and move the window forward to repeat till end of signal.


18c.bmp (1366 x 652 = 0.89 MP , 2,611 KB) [26 / 44] 100%
DUALITY
Time Domain
Amplitude
Time in seconds
Frequency Domain
Amplitude
Frequency in Hertz i.e. cycles/second
Time Domain
Amplitude
Time in seconds
Frequency Domain
Amplitude
Frequency in Hertz i.e. cycles/second
Time Domain
Amplitude
Time in seconds
Frequency Domain
Amplitude
Frequency in Hertz i.e. cycles/second
© Manish Kashyap

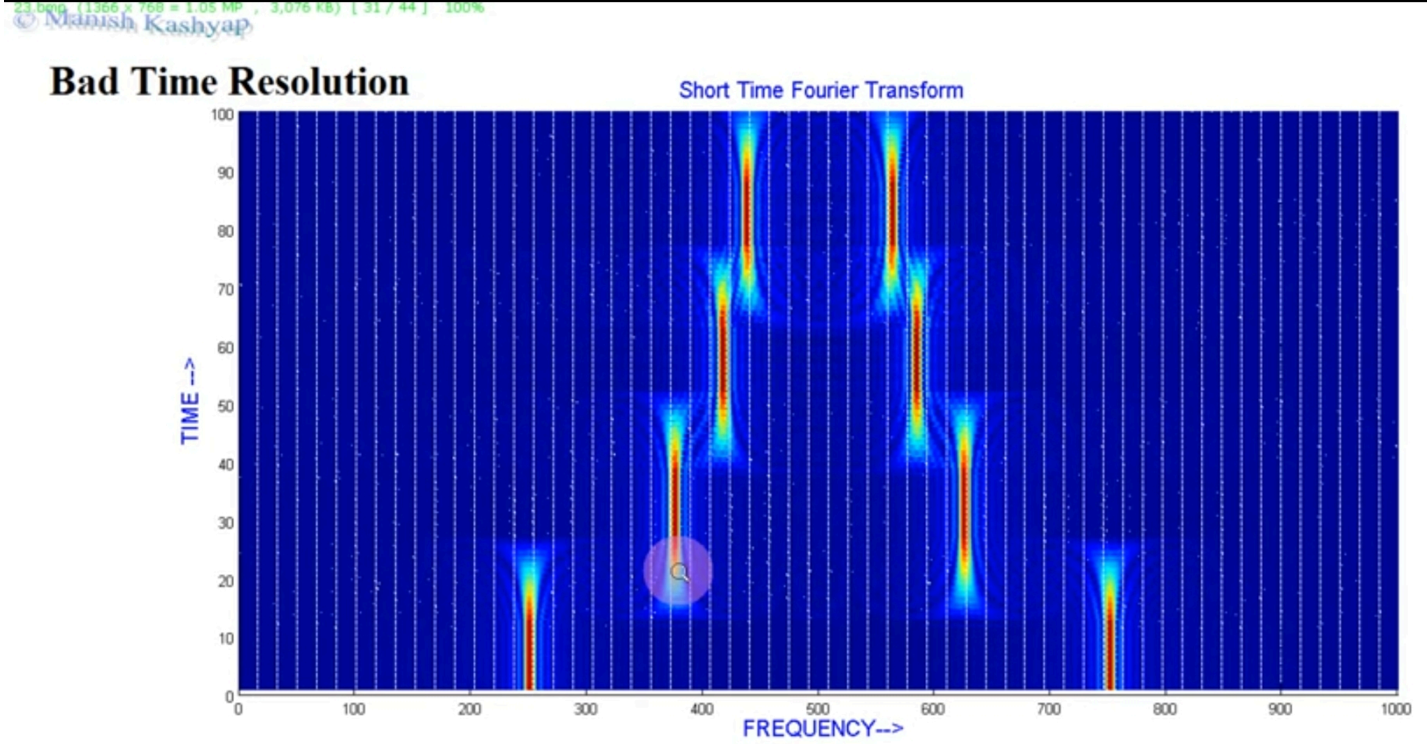

19.bmp (1366 x 651 = 0.89 MP , 2,607 KB) [27 / 44] 100%
Bad Time Resolution
DUALITY
Good Frequency Resolution
Time Domain
Amplitude
1
0.5
0
-0.5
-1
0
100
200
300
400
500
600
700
800
900
1000
Time in seconds
© Manish Kashyap


20.bmp (1366 x 651 = 0.89 MP , 2,607 KB) [28 / 44] 100%
Good Time Resolution
Short Time Fourier Transform
AMPLITUDE
0.03
0.02
0.01
0
TIME -->
FREQUENCY-->
0
100
200
300
400
500
600
700
800
900
1000
10
20
30
40
50
60
70
80
90
100
© Manish Kashyap

Universitat Pompeu Fabra Barcelona

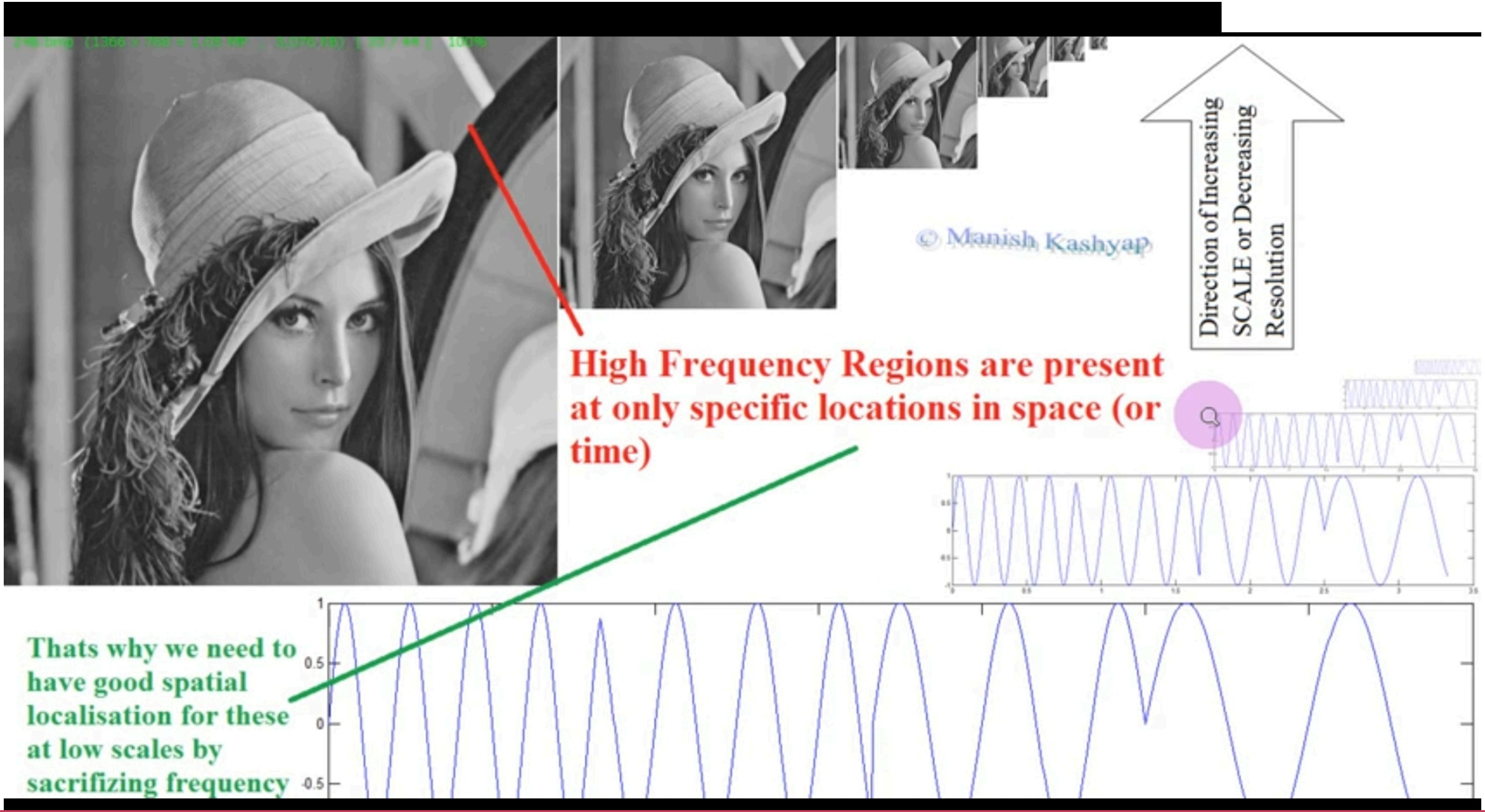


Universitat Pompeu Fabra Barcelona



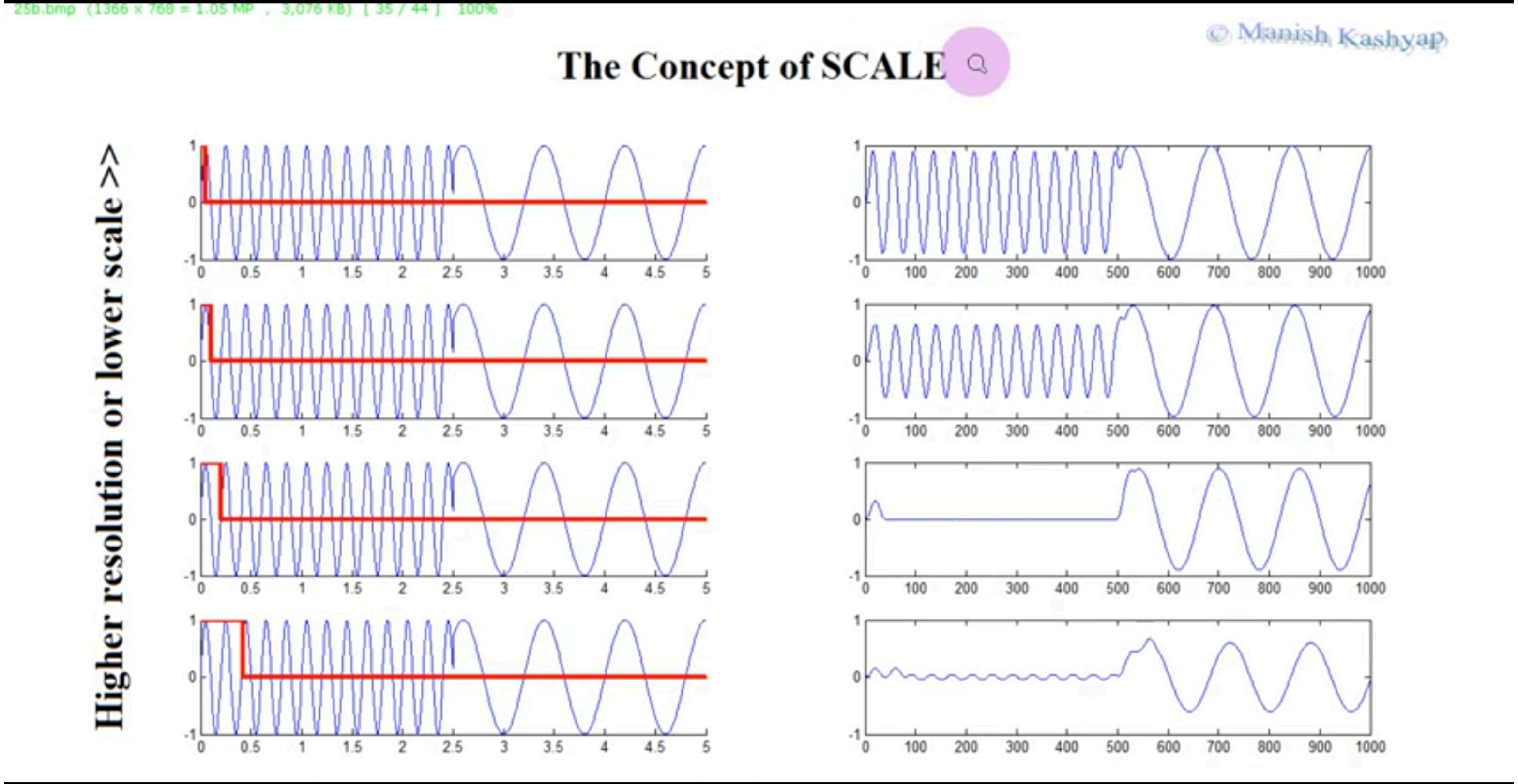
Universitat Pompeu Fabra Barcelona

The problem in resolution of STFT gets solved with fixed window size

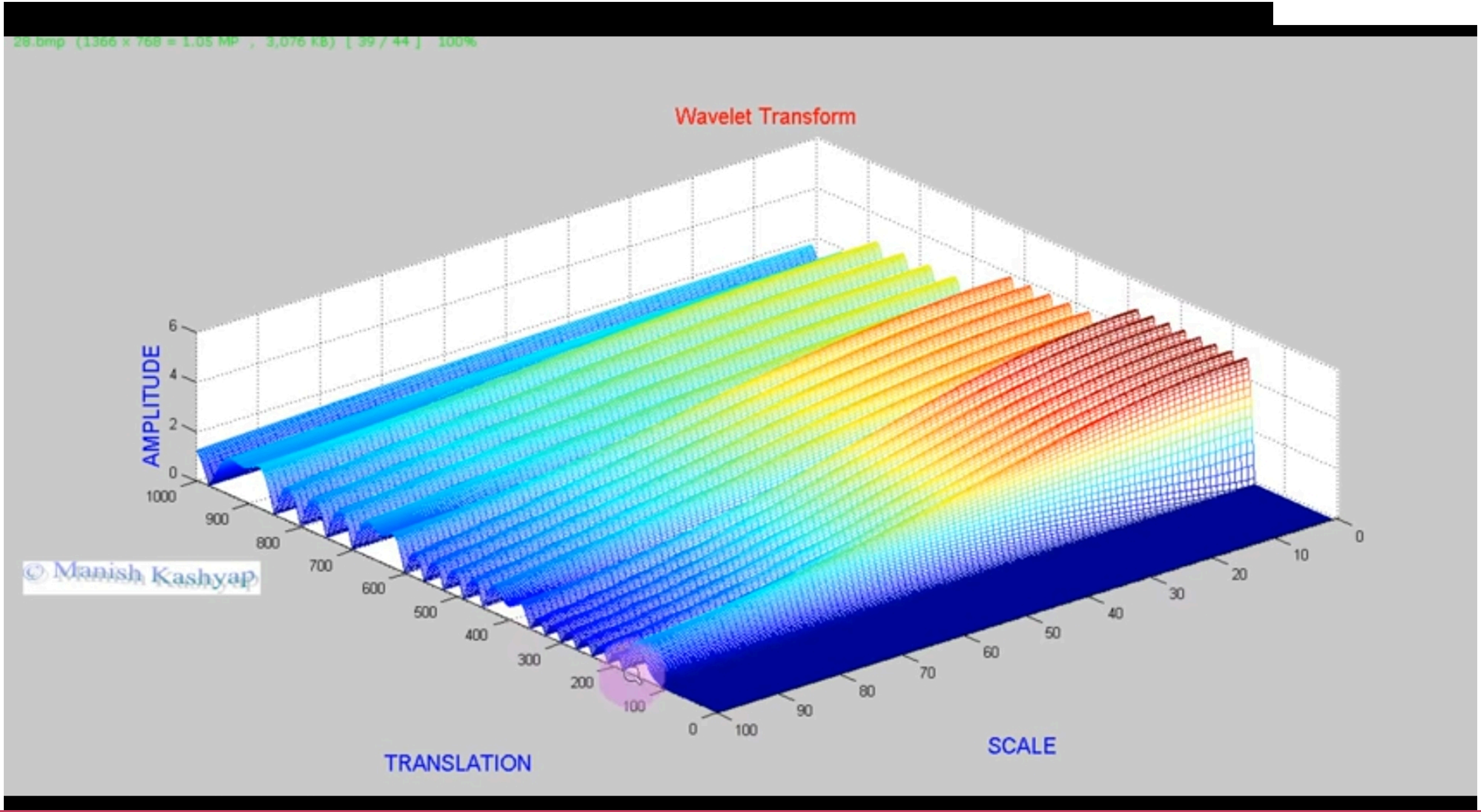


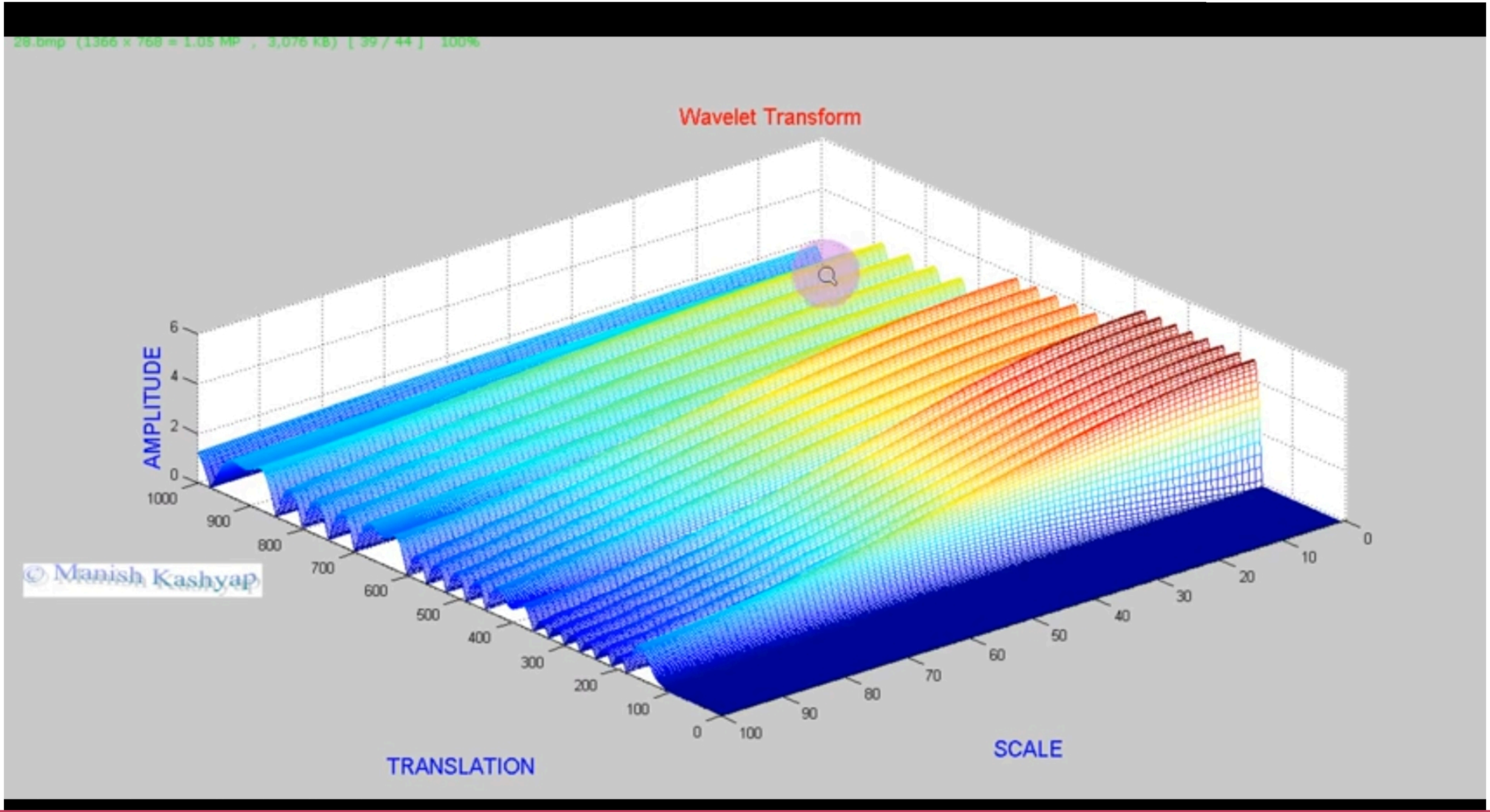
© Manish Kashyap
Direction of Increasing SCALE or Decreasing Resolution
High Frequency Regions are present at only specific locations in space (or time)
Thats why we need to have good spatial localisation for these at low scales by sacrifizing frequency
1
0.5
0
-0.5

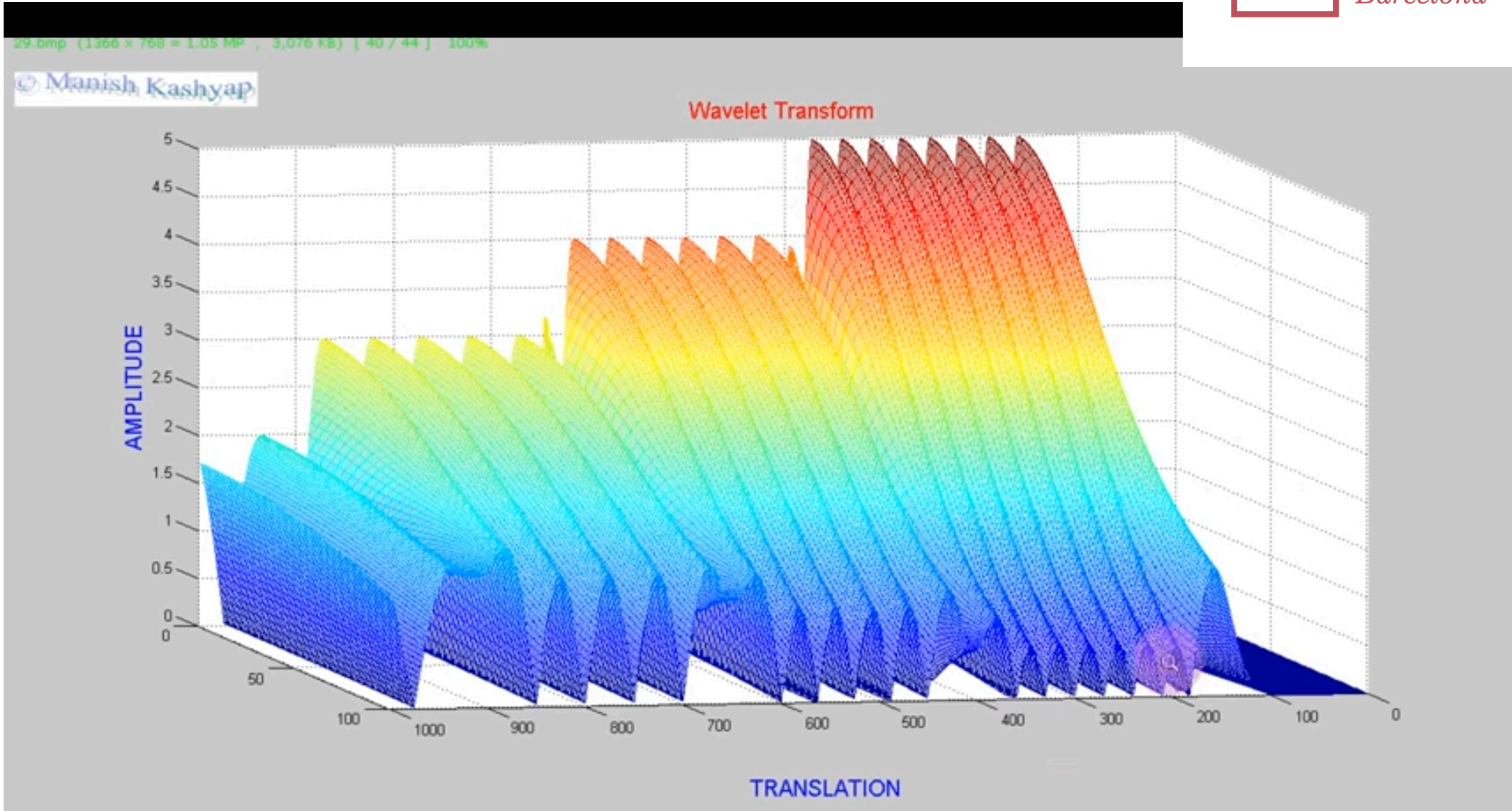
Wavelet transform

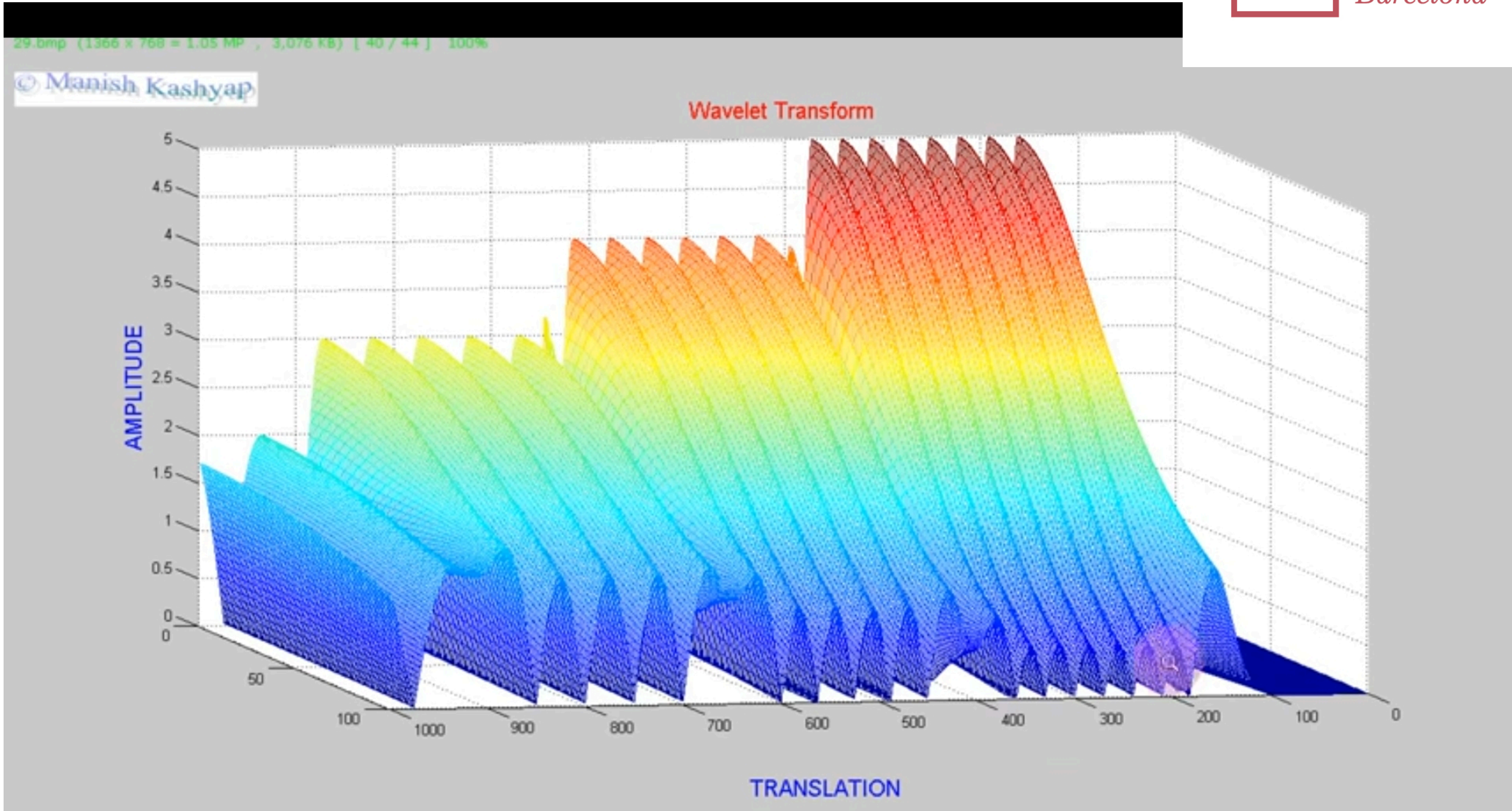

25b.bmp (1366 x 768 = 1.05 MP , 3,076 KB) [35 / 44] 100%
© Manish Kashyap
The Concept of SCALE
Higher resolution or lower scale >>
1
0
-1
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5
0 100 200 300 400 500 600 700 800 900 1000

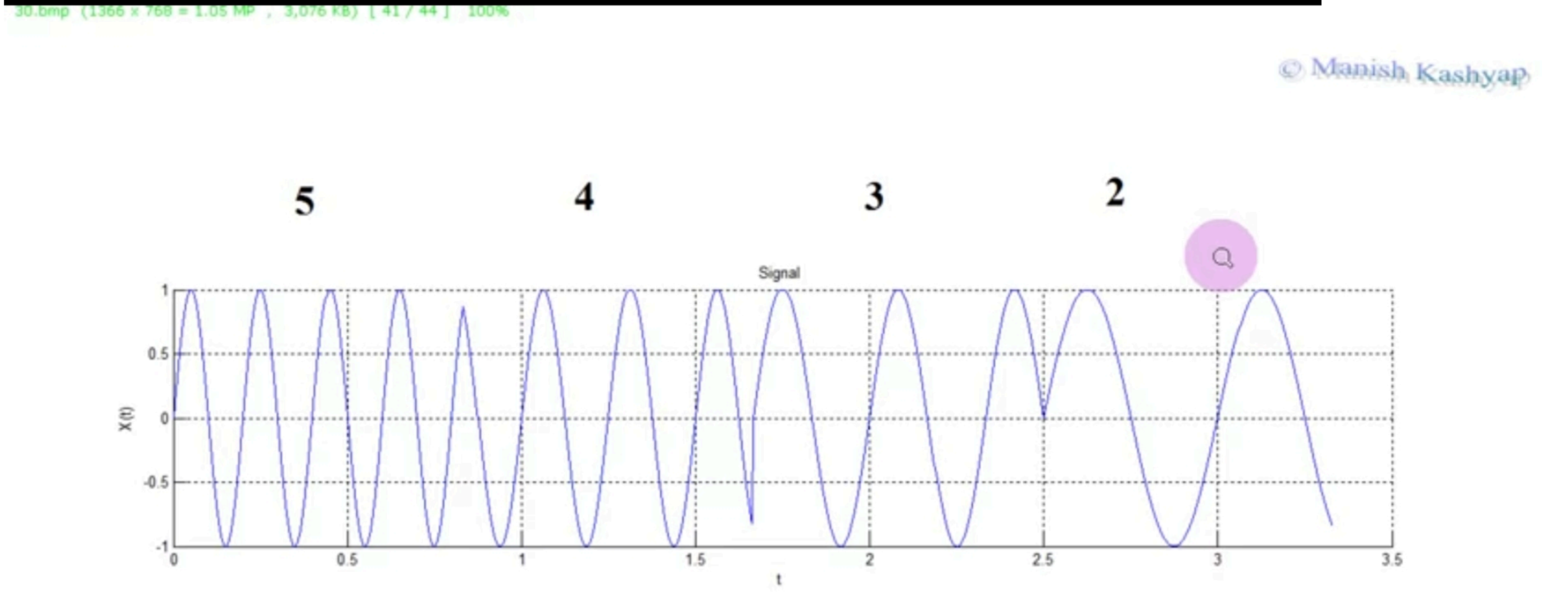
Universitat Pompeu Fabra Barcelona



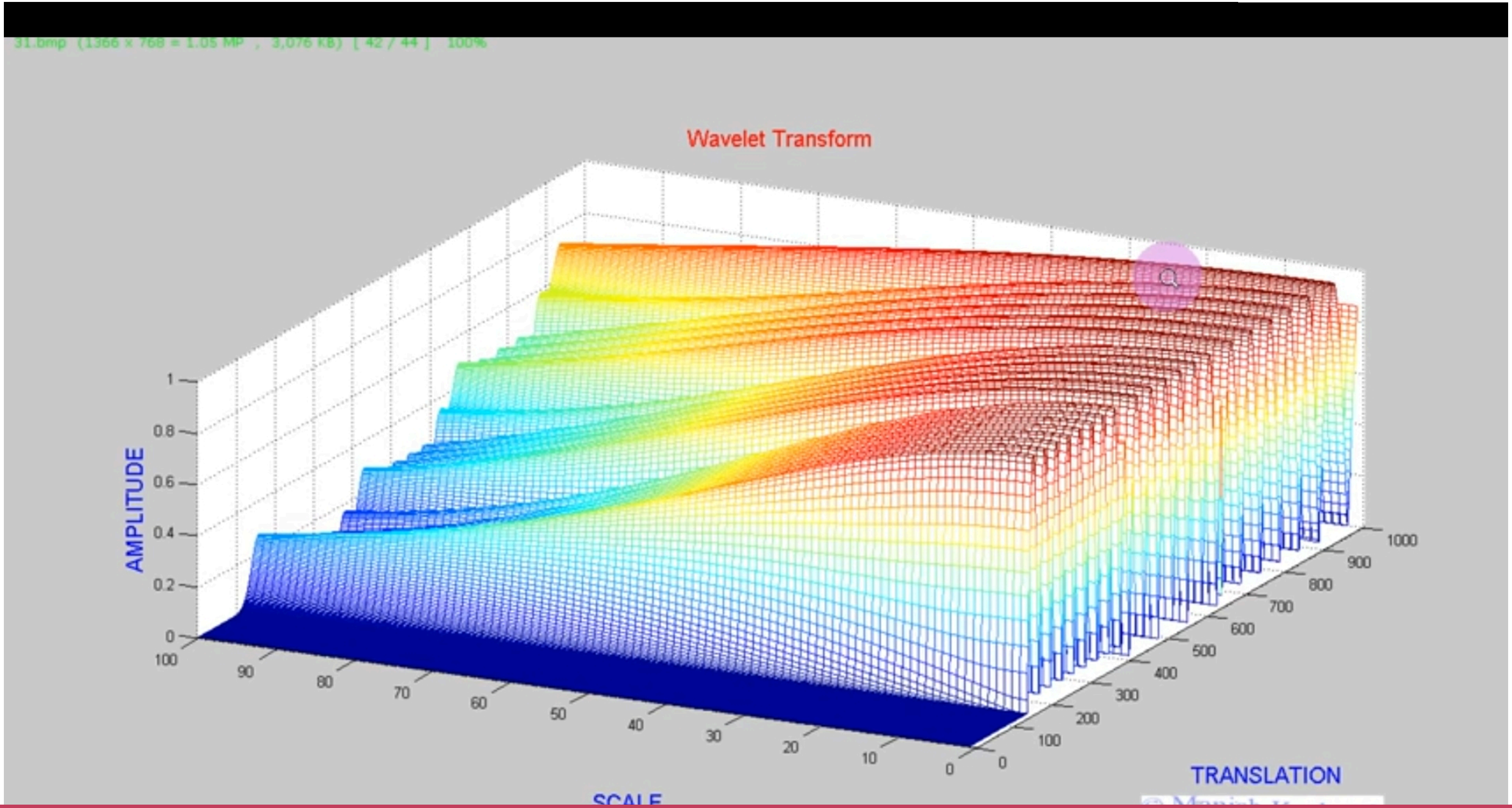

Wavelet Transform
AMPLITUDE
0
2
4
6
TRANSLATION
0
100
200
300
400
500
600
700
800
900
1000
SCALE
0
10
20
30
40
50
60
70
80
90
100
© Manish Kashyap


© Manish Kashyap
Wavelet Transform
AMPLITUDE
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
50
100
1000
900
800
700
600
500
400
300
200
100
0
TRANSLATION


© Manish Kashyap
Wavelet Transform
AMPLITUDE
TRANSLATION
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
50
100
1000
900
800
700
600
500
400
300
200
100
0

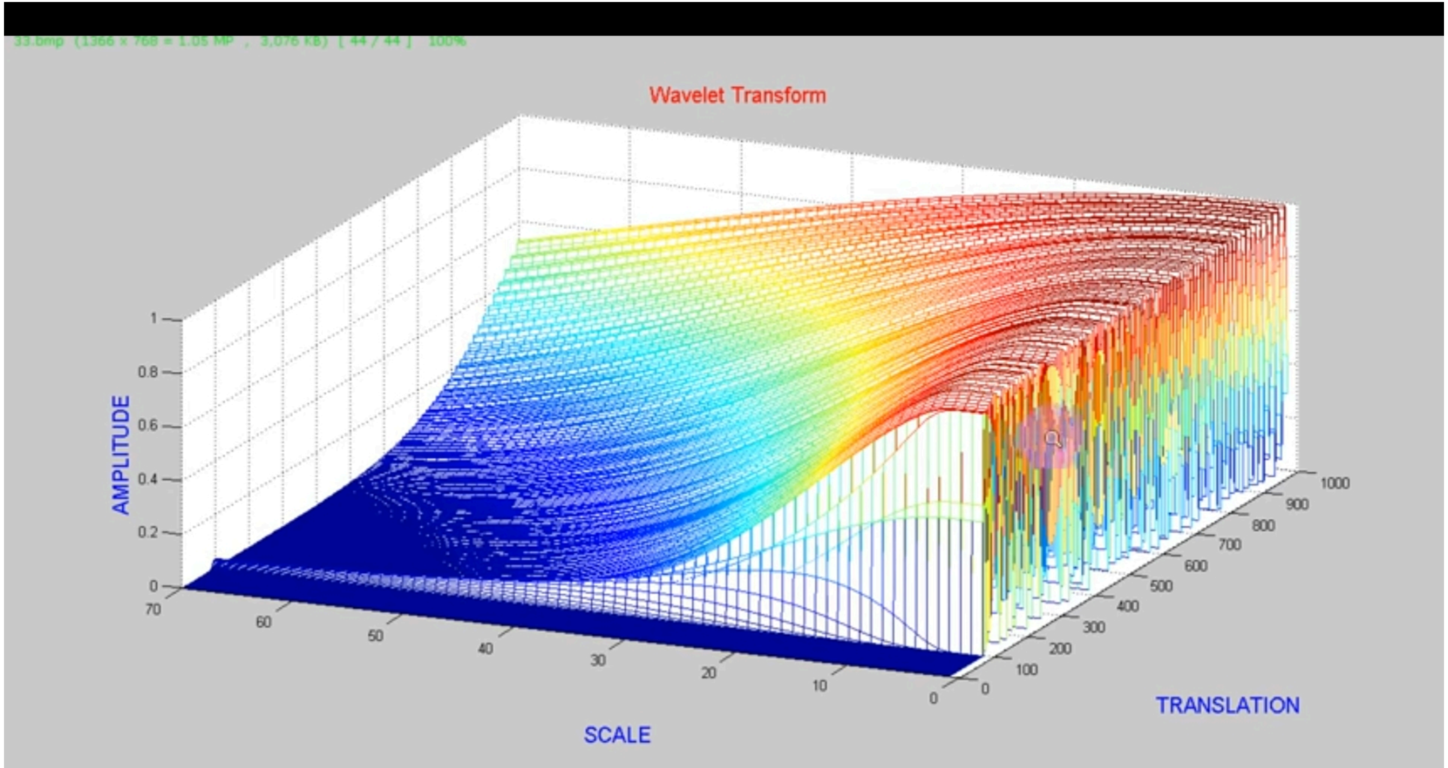

30.bmp (1366 x 768 = 1.05 MP , 3,076 KB) [41 / 44] 100%
© Manish Kashyap
5
4
3
2
Signal
X(t)
1
0.5
0
-0.5
-1
0
0.5
1
1.5
2
2.5
3
3.5
t

Universitat Pompeu Fabra Barcelona



Universitat Pompeu Fabra Barcelona

Example with chirp signal



Comparision between 2 standards

Figure 3.12: Recovered images after JPEG compression with ratios $k = 1, 5, 10, 20$.

Comparision between 2 standards

Figure 5.2: Lenna's image compressed using JPEG 2000 (left) and JPEG (right) at rates 0.0625 (up), 0.125 (middle), and 0.25 (down).

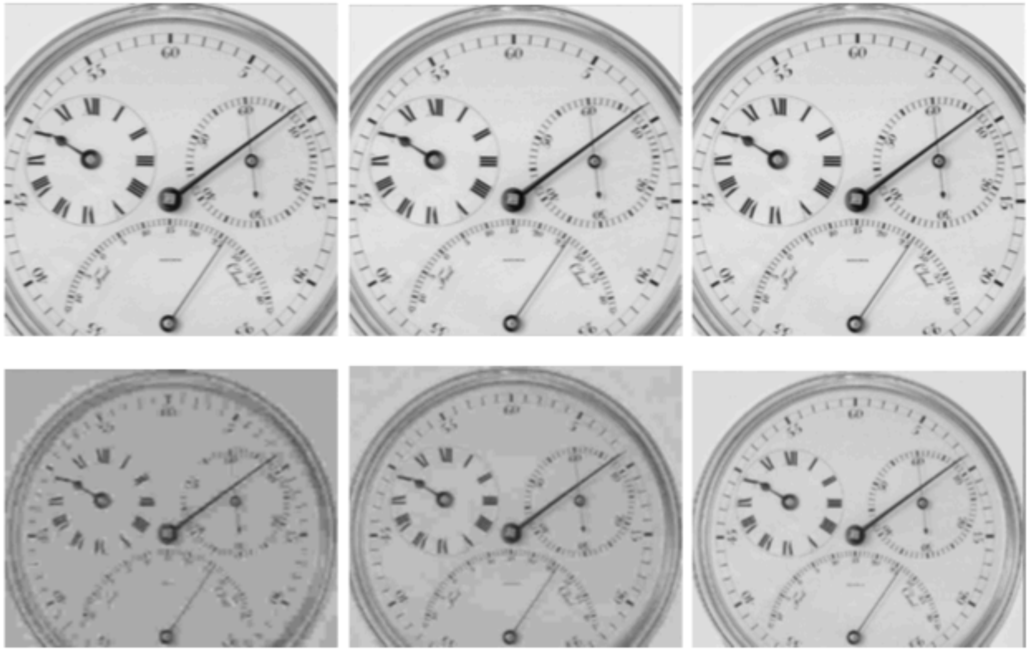


Figure 5.4: Chronometer image compressed using JPEG 2000 (up) and JPEG (down) at rates 0.0625 (left), 0.125 (middle), and 0.5 (right).

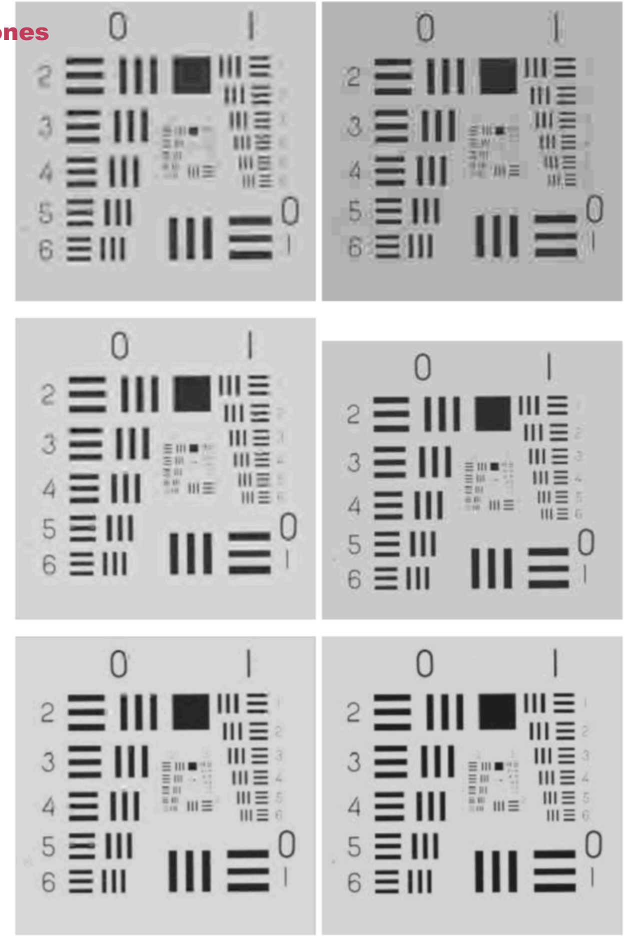


Figure 5.7: Pattern image compressed using JPEG 2000 (left) and JPEG (right) at rates 0.125 (up), 0.25 (middle), and 0.5 (down).

Other standards which seemed to success further

Graphical Image Interface

- File extension is .gif

- 8 bits per pixel for each image, up to 256 different colors chosen from the 24-bit RGB color space

(Remember this slide?)

Universitat Pompeu Fabra Barcelona

Other ways to encode a color image

Many other possible models may be used to represent the colors that make up an image. We could, for instance, use an indexed palette where we'd only need a single byte to represent each pixel instead of the 3 needed when using the RGB model. In such a model we could use a 2D matrix instead of a 3D matrix to represent our color, this would save on memory but yield fewer color options.

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F

Graphical Image Interface

- Supports animations
- Supports separate palette of up to 256 colors for each frame.

Graphical Image Interface

·Less suitable for color photographs, enough for graphics or logos

·GIF images are compressed using the Lempel–Ziv–Welch (LZW) lossless data compression technique to reduce the file size without degrading the visual quality.

Graphical Image Interface

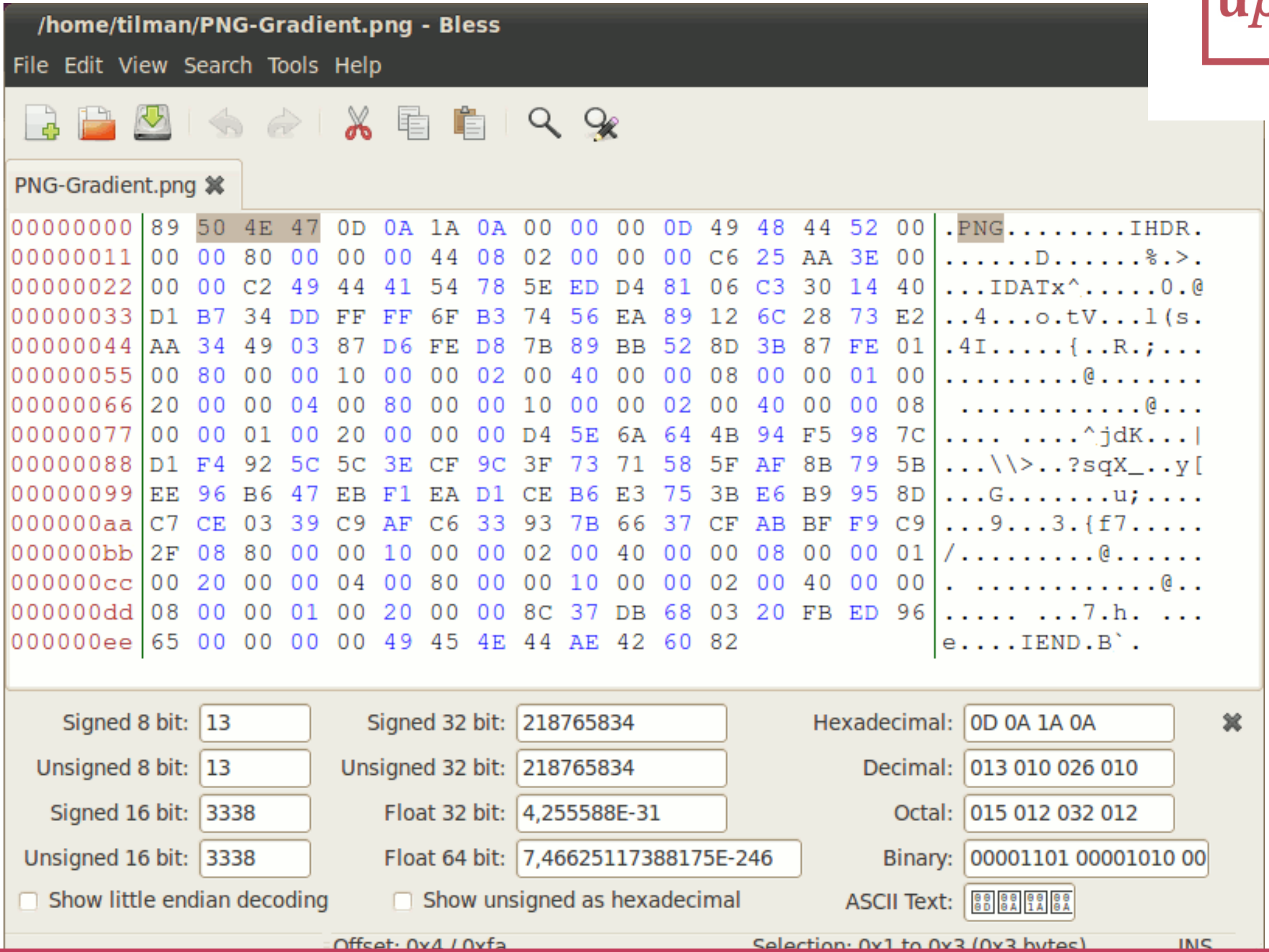
·Very common on the internet, specially 4chan, Forocoches, Twitter and any website/social media full of trolls

Portable Network Graphics (.png)

- Non-patented, designed to substitute .gif
- Supports 24RGB or 32 bit RGBA
- Lossless codec, based on DEFLATE encoding

Universitat Pompeu Fabra Barcelona

(DEFLATE encoding: a mix of LZSS and Huffman)


/home/tilman/PNG-Gradient.png - Bless
File Edit View Search Tools Help
PNG-Gradient.png
00000000 89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52 00 .PNG........IHDR.
00000011 00 00 80 00 00 00 44 08 02 00 00 00 C6 25 AA 3E 00 ......D......%.>.
00000022 00 00 C2 49 44 41 54 78 5E ED D4 81 06 C3 30 14 40 ...IDATx^.....0.@
00000033 D1 B7 34 DD FF FF 6F B3 74 56 EA 89 12 6C 28 73 E2 ..4...o.tV...l(s.
00000044 AA 34 49 03 87 D6 FE D8 7B 89 BB 52 8D 3B 87 FE 01 .4I.....{..R.;...
00000055 00 80 00 00 10 00 00 02 00 40 00 00 08 00 00 01 00 ....@.......
00000066 20 00 00 04 00 80 00 00 10 00 00 02 00 40 00 00 08 ....@...
00000077 00 00 01 00 20 00 00 00 D4 5E 6A 64 4B 94 F5 98 7C ....^jdK...|
00000088 D1 F4 92 5C 5C 3E CF 9C 3F 73 71 58 5F AF 8B 79 5B ...\\>..?sqX_..y[
00000099 EE 96 B6 47 EB F1 EA D1 CE B6 E3 75 3B E6 B9 95 8D ...G.......u;....
000000aa C7 CE 03 39 C9 AF C6 33 93 7B 66 37 CF AB BF F9 C9 ...9...3.{f7.....
000000bb 2F 08 80 00 00 10 00 00 02 00 40 00 00 08 00 00 01 /.........@......
000000cc 00 20 00 00 04 00 80 00 00 10 00 00 02 00 40 00 00 . ...@..
000000dd 08 00 00 01 00 20 00 00 8C 37 DB 68 03 20 FB ED 96 ....7.h. ...
000000ee 65 00 00 00 00 49 45 4E 44 AE 42 60 82 e....IEND.B`.
Signed 8 bit: 13
Unsigned 8 bit: 13
Signed 16 bit: 3338
Unsigned 16 bit: 3338
Signed 32 bit: 218765834
Unsigned 32 bit: 218765834
Float 32 bit: 4,255588E-31
Float 64 bit: 7,46625117388175E-246
Hexadecimal: 0D 0A 1A 0A
Decimal: 013 010 026 010
Octal: 015 012 032 012
Binary: 00001101 00001010 00
ASCII Text:
Show little endian decoding
Show unsigned as hexadecimal
Offset: 0x4 / 0xfa
Selection: 0x1 to 0x3 (0x3 bytes)
INS

Bonus track: if you want to know exactly how an image is get from the real world:

https://www.cambridgeincolour.com/tutorials/camera-sensors.htm

Thanks

franciscojavier.brines@upf.edu