FUNDAMENTALS OF IMAGE AND VIDEO PROCESSING

Part 3: From 1D to m-D signals

What we'll see in this section

- From 1-D to m-D signals (images)
  - Acquisition process
  - Mathematical representation in space and frequency domains
- Multi-dimensional analog systems
  - Extension of impulse response and convolution
- Analog to digital conversion
  - Sampling in 2D
  - Quantization of visual signals
  - Color representations
- Multi-dimensional digital systems
  - Discrete 2D convolution
  - 2D DFT and frequency domain digital filtering

Multi-dimensional signals

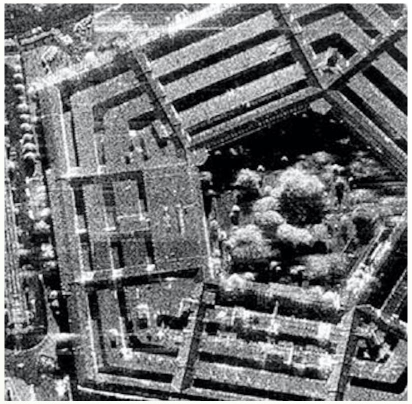
- Some signals require more than one dimension to be expressed:
  - Classical pictorial images require 2 dimensions → i(x,y)
  - Volumetric data (e.g., CAT, NMR) require 3 dimensions → d(x,y,z)
  - Videos require a mixed space-time 3D domain → v(x,y,t)
  - Moving point-clouds require 4 dimensions → c(x,y,z,t)
- Some elements of the theory we've just seen need to be adapted to deal with these more complex domains

- *By now, we will focus on **images** (2D signals in the space domain)*
  - *Extension to 3D space domain (x,y,z) is rather trivial*
  - *Extension to mixed time-space domain will be introduced later*

Image acquisition

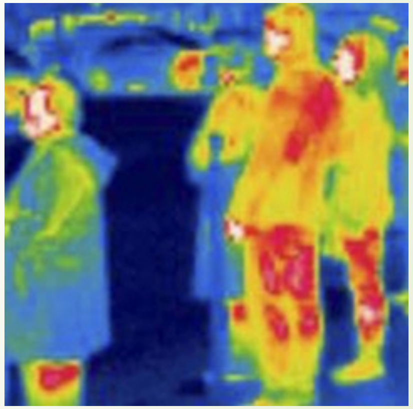
- An image is typically generated by an acquisition device, which translates a m-D physical stimulus into a 2D signal
- The stimulus can refer to any physical quantity
  - An appropriate sensor is required to convert that specific physical phenomenon into an electrical signal

Optical image



SAR image

X-ray image



Thermal image

Acquisition process

- Independently of the physical phenomenon to measure, the acquisition process is more or less similar, and it is made of:
  - An **image formation system**: projects the desired snapshot of the m-D reference world into the 2D image plane
  - A **sensor**: translates the physical quantity to be measured within the image plane into an electrical signal
  - A **recorder**: stores the acquired signal into some physical device
- Depending on the physical nature of the signal to be acquired we will have different hardware components
  - E.g., optical sensor will be sensitive to visible light radiations, thermal sensors will be sensitive to infrared radiations, etc.

Acquisition process: example

Sensor:
- CCD or CMOS

Recorder:
- RAM
- Flash memory

Image formation:
- Lenses → focus
- Diaphragm → aperture
- Shutter → exposition

Acquisition vs. human sensing

- Humans are able to perceive a subset of possible physical stimula according to their (limited) senses, machines can do more
  - E.g., we can perceive images into the visible range of wavelengths, but we are unable to perceive infrared or ultraviolet
- Human sensing mechanisms are quite similar to artificial ones

Image signals vs. math

- Let's consider a grey level picture

- It can be seen as a **continuous signal i(x,y)**, whose amplitude is given by the luminance value at coordinates x,y

Images as 2D surfaces

- An image can be also seen as a surface in the 2D space
  - Sometimes it will be useful to look at it in that way

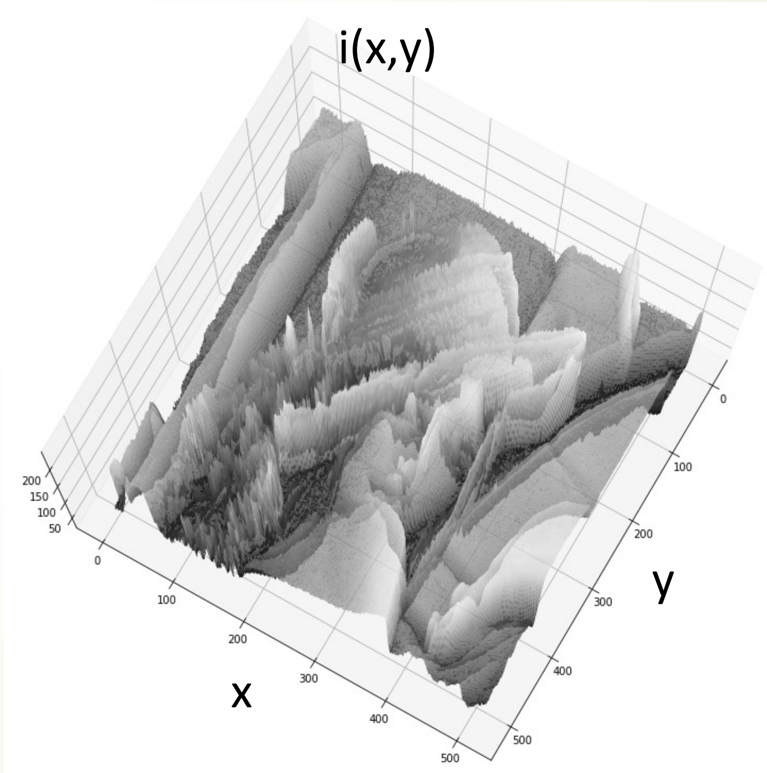
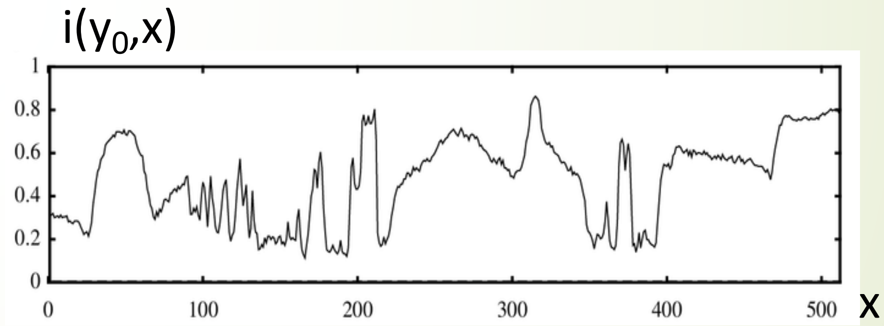


Image scan lines

- A simpler and sometimes useful way to represent an image signal is to extract scan lines (intensity profiles over a row or column)



2D signals in the frequency domain

- Also in this case, frequency representation is often useful
  - Fourier transform straightforwardly extends to the 2D domain:

$$I(f_x, f_y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} i(x,y) e^{-j2\pi(f_x x + f_y y)} dx\, dy$$

DIRECT 2D FT

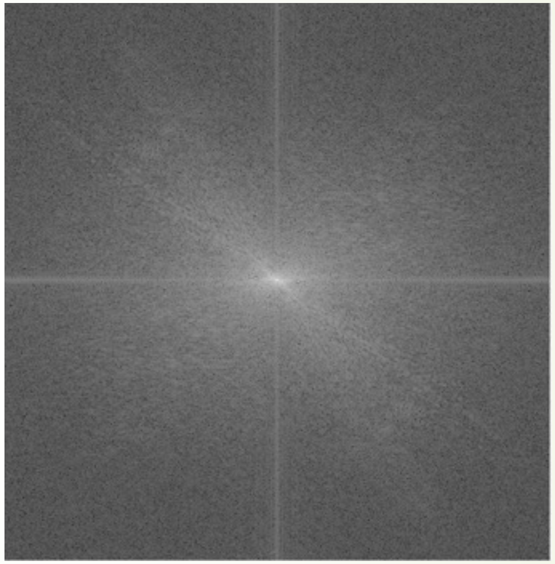
$$i(x,y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} I(f_x, f_y) e^{j2\pi(f_x x + f_y y)} df_x df_y$$

INVERSE 2D FT

NB. We are implicitly saying that the 2D signal is represented by a combination of horizontal (f_x) and vertical (f_y) frequency components

Image spectrum

- Spatial vs. frequency representation



  - The 2D spectrum makes evident the low-pass nature of the image
  - This characteristic is very common in natural images, and is associated to concepts such as smoothness, correlation, slow variation

2D systems (filters)

- To process multi-dimensional signals we need multi-dimensional systems. In the 2D case:

$$f(i(x,y)) = \hat{\imath}(x,y)$$

- Linearity and domain invariancy are again important properties
  - A 2D system is called **LSI** (linear, space-invariant) if it fulfills both superposition and shift properties:

$$\forall i(x,y): f(i(x,y)) = \hat{\imath}(x,y)$$
$$\forall j(x,y): f\big(j(x,y)\big) = \hat{\jmath}(x,y)$$
$$f\big(\alpha\, i(x,y) + \beta\, j(x,y)\big) = \alpha\hat{\imath}(x,y) + \beta\hat{\jmath}(x,y);\ \forall \alpha,\beta$$

superposition

$$\forall i(x,y): f\big(i(x,y)\big) = \hat{\imath}(x,y)$$
$$f(i(x-x_0, y-y_0)) = \hat{\imath}(x-x_0, y-y_0)$$

shift invariance

Response of m-D LSI systems

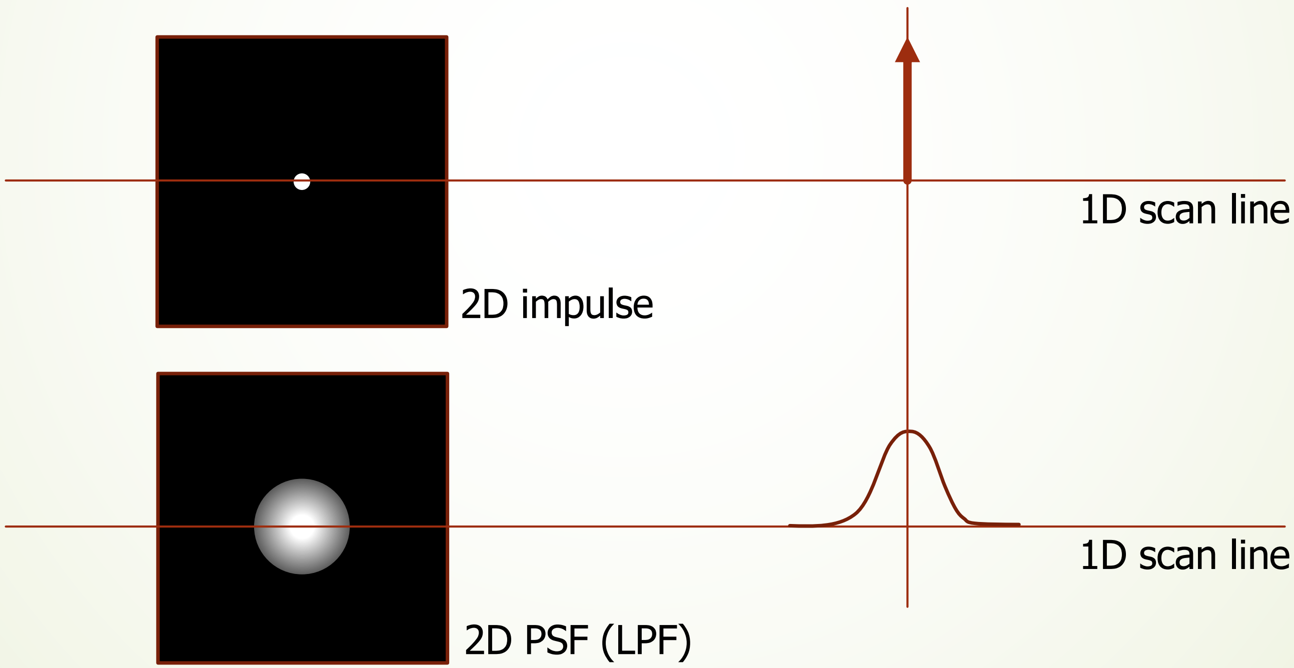
- Exactly the same way a 1D LTI system outputs the convolution between the input and its impulse response, an m-D system responds with a **m-D convolution**
- Skipping the proof (analogous to the 1D case), in 2D we have:

$$\hat{\imath}(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} i(\lambda,\mu)\cdot h(x-\lambda, y-\mu)\,\partial\lambda\,\partial\mu = i(x,y) * h(x,y)$$

- h(x,y) is the 2D impulse response, i.e., the way our system reacts to a 2D impulse in input
- Since a 2D impulse can be though as a point in space, and the system typically expands that point in some way, the 2D impulse response takes the name of **Point Spread Function** (PSF)

PSF: example

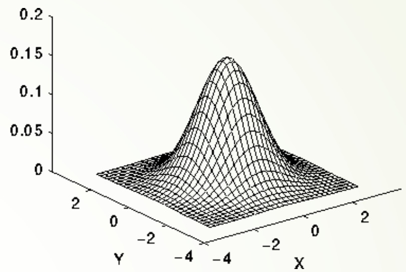
- We can see an impulse in space as a white dot on a black background (black being the absence of signal)
- PSF is the 2D output when an impulse is input. Its shape depends on the system characteristics



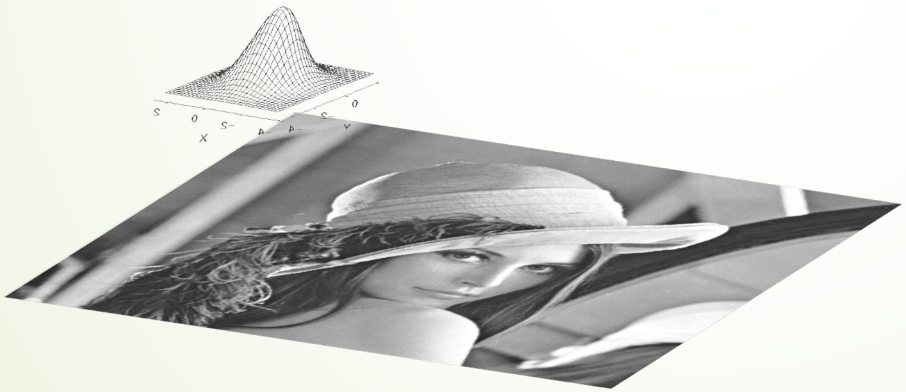
2D convolution: how it works

input image

*



filter response (PSF)



convolution

output image

And now… let's go digital

- Also in the case of image signals, if we want to manipulate them with a computer we need to convert them in binary form
- As for 1D signals, we have to perform analog to digital (AD) conversion
  - Once again, we need to apply **sampling, quantization and coding** in sequence
  - Let see how those operations work in the case of images

Image sampling

- Images are theoretically unbound in resolution
  - You can zoom and zoom and every time you'll find new details…

So… how do we determine Nyquist frequency for images?

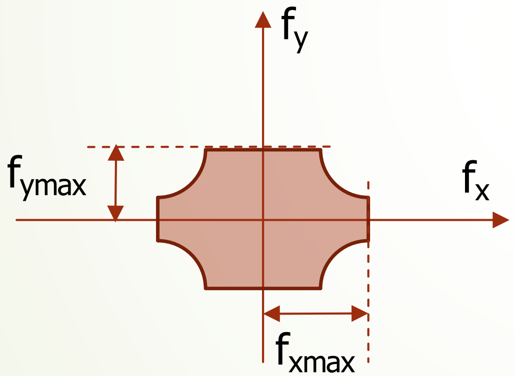
- The maximum frequency in a captured image is not a property of the image itself, but it is enforced by the acquisition system
  - Acquisition systems (e.g., photo cameras) act as **low-pass filters**
- We can determine the Nyquist frequency by Fourier transforming the continuous-space captured image
  - Since we are in 2D, we'll get two cut-off frequencies, on horizontal and vertical directions, respectively (f_{xmax}, f_{ymax})
  - To avoid aliasing, Nyquist criterion should be applied to both

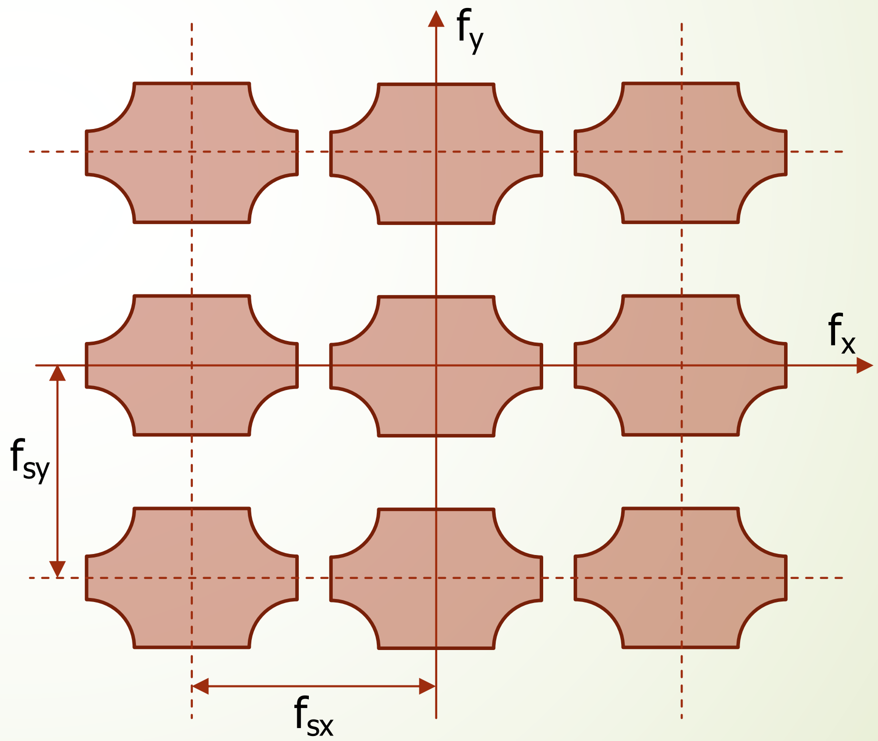
$$\boxed{f_{sx} \geq 2 \cdot f_{xmax} \quad ; \quad f_{sy} \geq 2 \cdot f_{ymax}}$$

NB. Although this may lead to different sampling frequencies in the two directions, typically we will use the same (the maximum one) in order to achieve a square grid.

Sampling

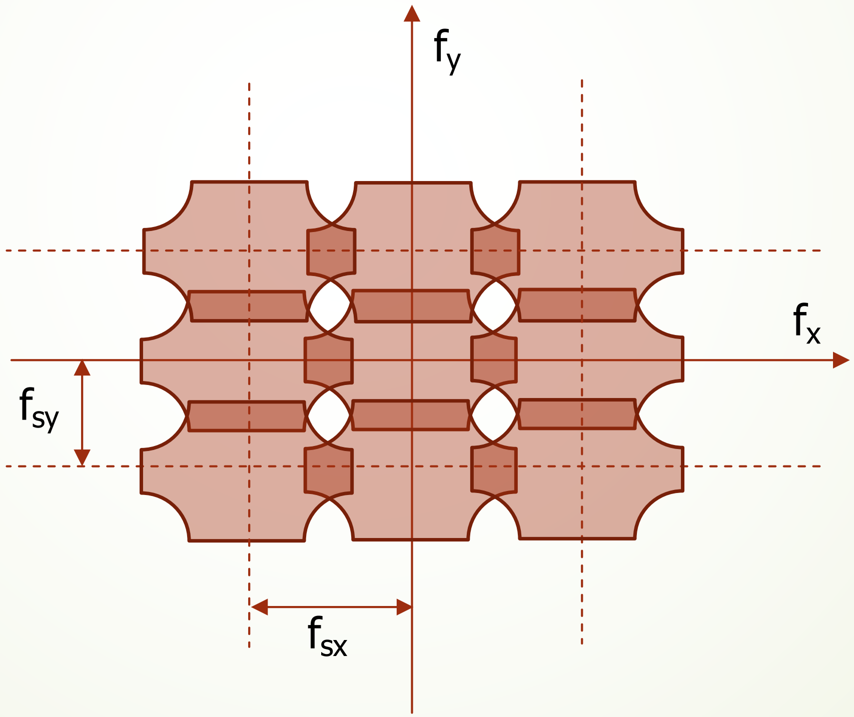
- Without repeating all the theory, ideal sampling will generate replicas of the spectrum in the two frequency directions
  - If we fulfill the Nyquist criterion, replicas will not overlap





Under-sampling and aliasing

- Also in this case, if the sampling frequency is too low (under-sampling) aliasing will appear



Example of aliasing visual effect

- Right image is sub-sampled below Nyquist rate

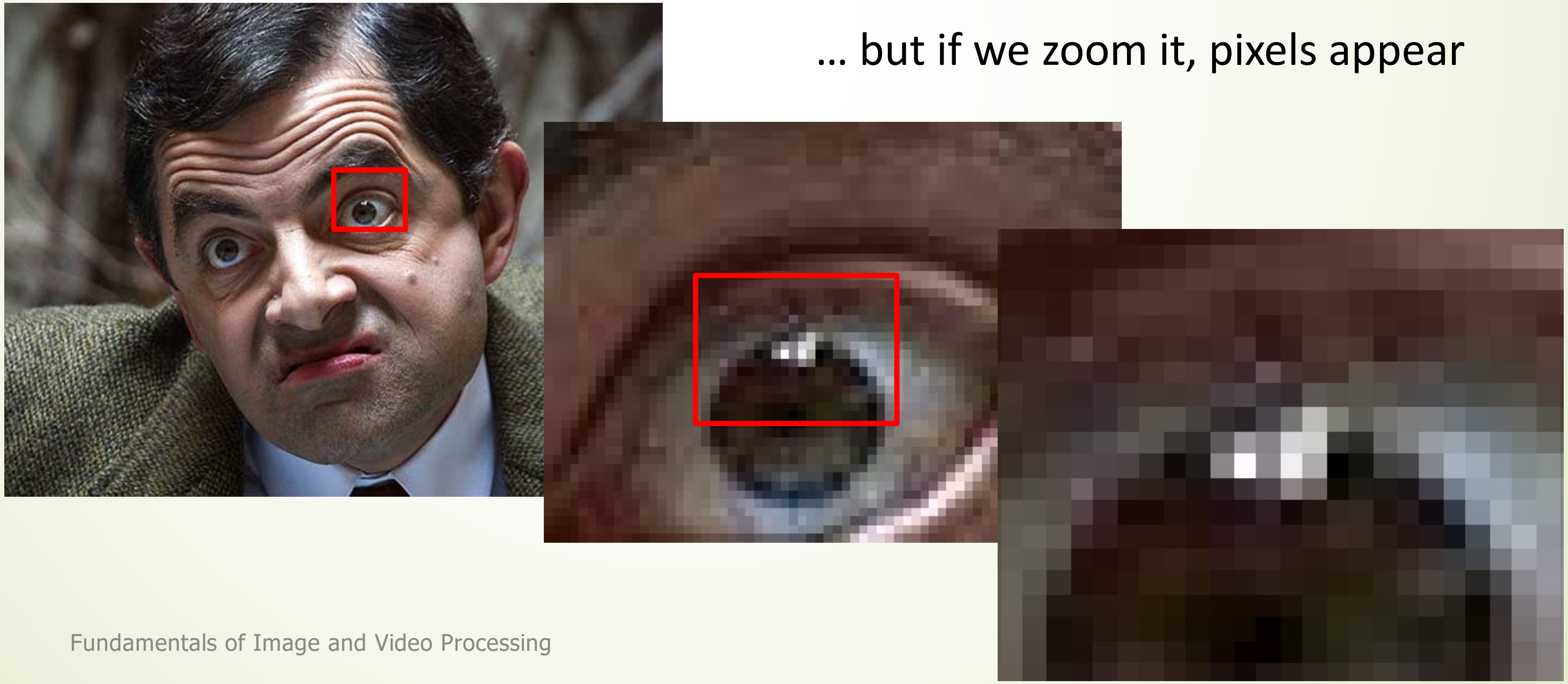
Source: Scientific Volume Imaging, svi.nl

NB. To avoid aliasing in image sub-sampling pre-filtering is required

Pixels

- After sampling, the image is converted into a discrete 2D collection of samples over an ordered grid (a **NxM matrix**)
- Each sample is called **pixel**, from the acronym of the words "**PIC**ture" "**EL**ement"
  - When we look at digital images we don't perceive pixels because they are very close to each other (the **human eye acts as a LPF**)

… but if we zoom it, pixels appear



Aspect ratio

- The aspect ratio is the ratio between horizontal and vertical dimensions of an image (X:Y)
- Some typical aspect ratio values are:
  - 4:3 → old TV sets (SDTV, VGA)
  - 16:9 → newer TV sets (HDTV, widescreen)
  - 1.85:1 or 2.39:1 → cinema
- Larger aspect ratios provide more “immersive” experience

Image Size vs image Resolution

- There is often a confusion between size and resolution of images
- **Size** tells the number of pixels it is made of (rows x columns)
  - E.g., a 4Mpixel camera produces 2592x1520 images in 16:9 aspect ratio, corresponding to 3,939,840 pixels
- **Resolution** refers instead to the number of pixels per unit space (e.g., expressed in **DPI**, dot per inch)
  - Clearly, if the same area is acquired with a sensor with more pixels, also the resolution will be higher
- Resolution should be appropriately set depending on the application we target, for instance:
  - if I have to detect an object on an image, I need a resolution that provides a minimum given number of points for that object
  - If I have to print something on a given surface, I need a resolution that prevents perceiving the individual pixels (***pixelation***)

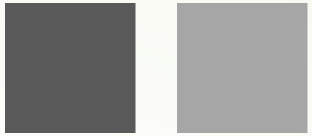
Image Size vs image Quality

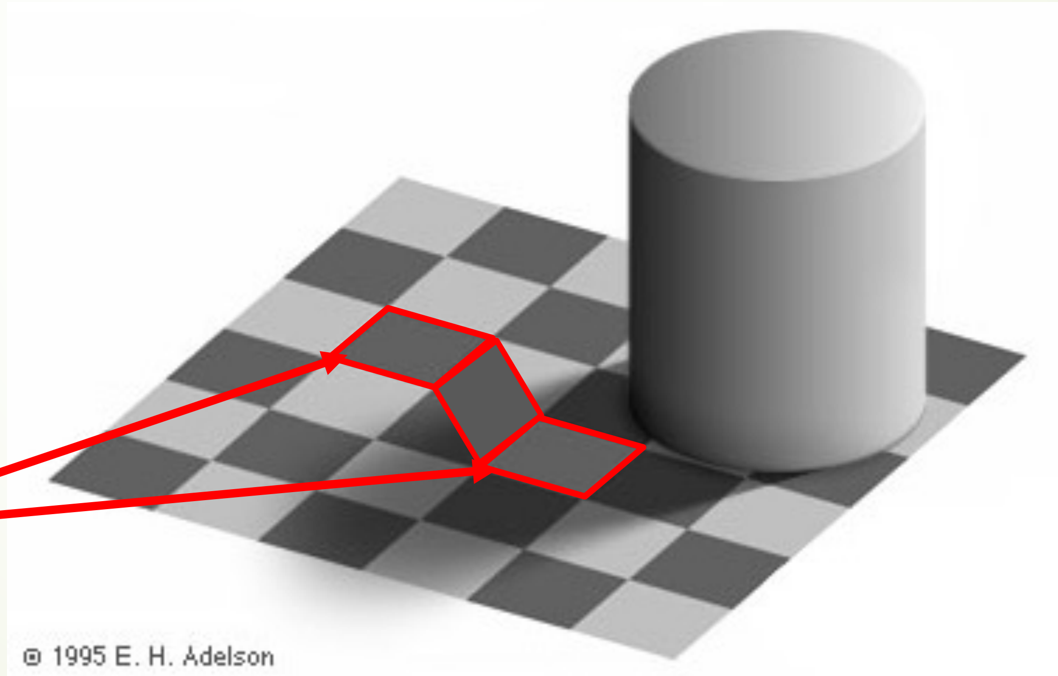
- Another common confusion is between image size and quality
  - Although having an insufficient number of pixels (with respect to the application) may hinder quality, having lots of pixels does not necessarily mean getting high-quality images
  - There are other important factors that determine quality:
    - **Optical components**: quality and size of lenses and mechanics are fundamental (blurring, distortion)
    - **Sensor**: quality and size of the sensor (CCD or CMOS) is also very important (low-pass effect, noise)

Pixel quantization

- Up to this point, pixel are represented by continuous values in a given range
- In order to complete the numerical conversion, we have to quantize them to achieve discrete amplitude values
  - In principle, quantization of pixel is completely equivalent to quantization of every other signal
  - In practice, we have to deal with the **perceptual impact** of the quantization on images

Math vs. Human perception

- Look at these two blocks: 
  - Everyone can see that the block on right is brighter
- but what about the following:

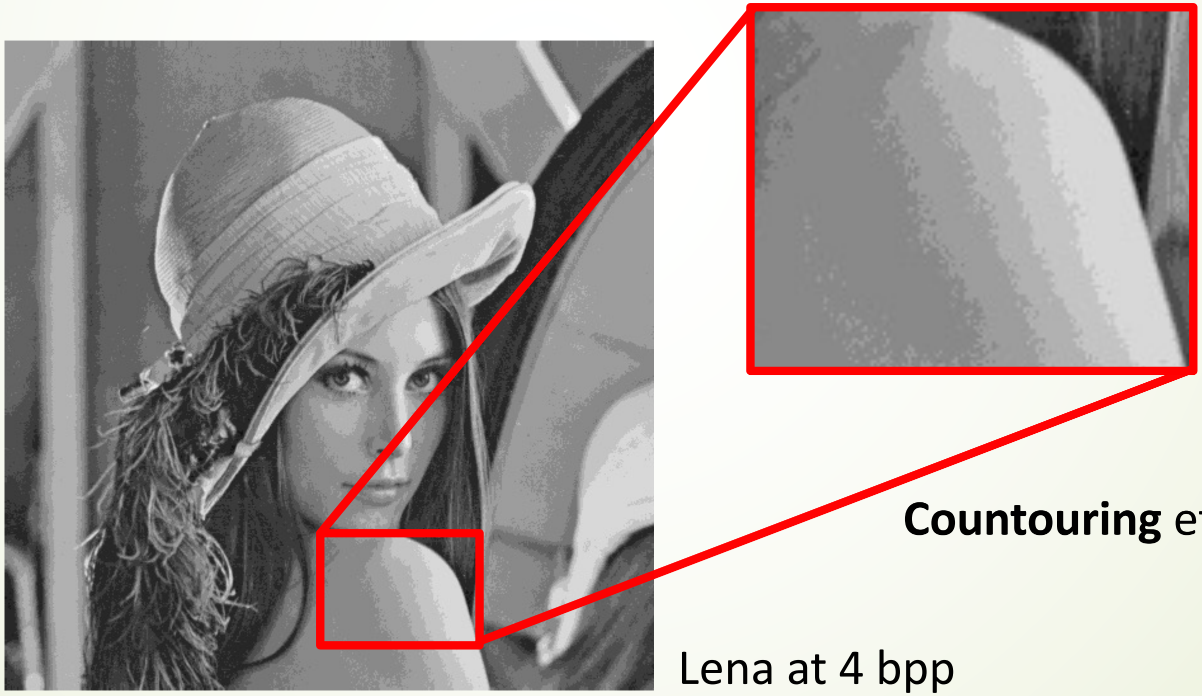


Which is brighter?

Adelson chessboard, ©1995

Uniform quantization

- Let consider a standard uniform quantizer at B **bit per pixel (bpp)**, applied to a greyscale image
- As we have seen in part 2, the SNR of our image will be **6B dB**
  - At 4 bpp we get a 24 dB image, which can be considered a low but nearly acceptable signal quality



Countouring effect appears!

Lena at 4 bpp

Contouring

- Contouring is due to the sensitivity of the **HVS (human visual system)** to contrasts
  - To avoid it we must be sure that transitions among quantized greylevels are below our perception threshold
  - If we double the number of bpp, thus having 256 grey levels, we are pretty sure that no contouring will be visible
    - *Incidentally, **8 bpp** means **1 byte per pixel** (perfect for memory alignment in computers) and around 50dB SNR (high-quality)*

Lena @ 8 bpp

Contrast quantization

- But… is it possible to go below 8 bpp while avoiding contouring?
  - The answer is 'yes', but we can't use a uniform quantizer
- The **Weber's law** says that the HVS is unable to perceive more than 50 contrast variations
  - More precisely, Weber says that the HVS is not able to distinguish greylevel variations that are below 2%, or:

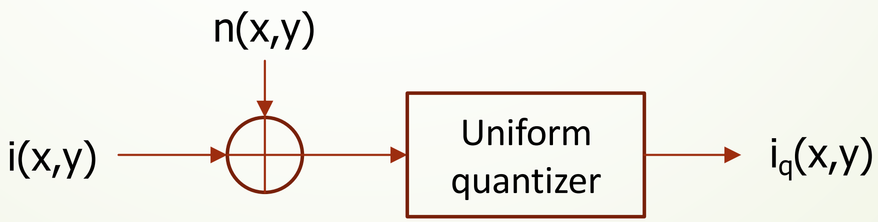
$$\frac{l_{fg}-l_{bg}}{l_{bg}} < 0.02$$

  - This means around 50 perceivable contrast levels (6 bits)
  - To **quantize the contrast** it is possible to apply a nonlinear transform (log) followed by uniform quantization

Example of stimulus used to test Weber's law

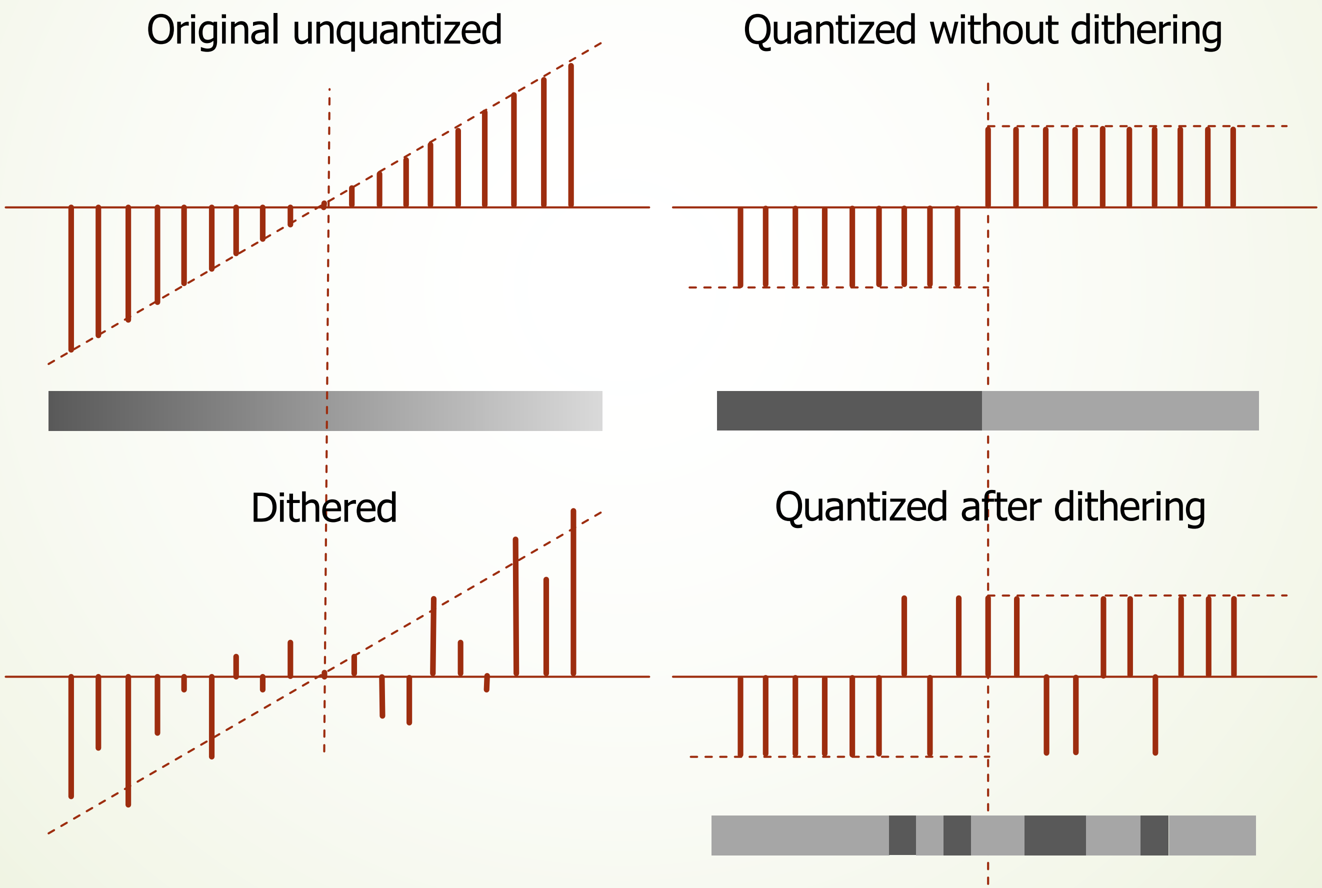
Dithering

- And if we want to go further? For instance quantize at 3-4 bpp without introducing contouring?
  - The answer is again 'yes', but we can't use a simple quantizer anymore
  - The idea is to fool the HVS with a 'trick'
- The technique is known as dithering: we add a pseudo-random noise n(x,y) (**dither**) to the image before quantizing it
  - The noise causes pixels nearby a greylevel transition to randomly switch their value with the neighboring one
  - Human eyes act as a lowpass filter, mixing the two values and virtually producing intermediate greylevels across the transition



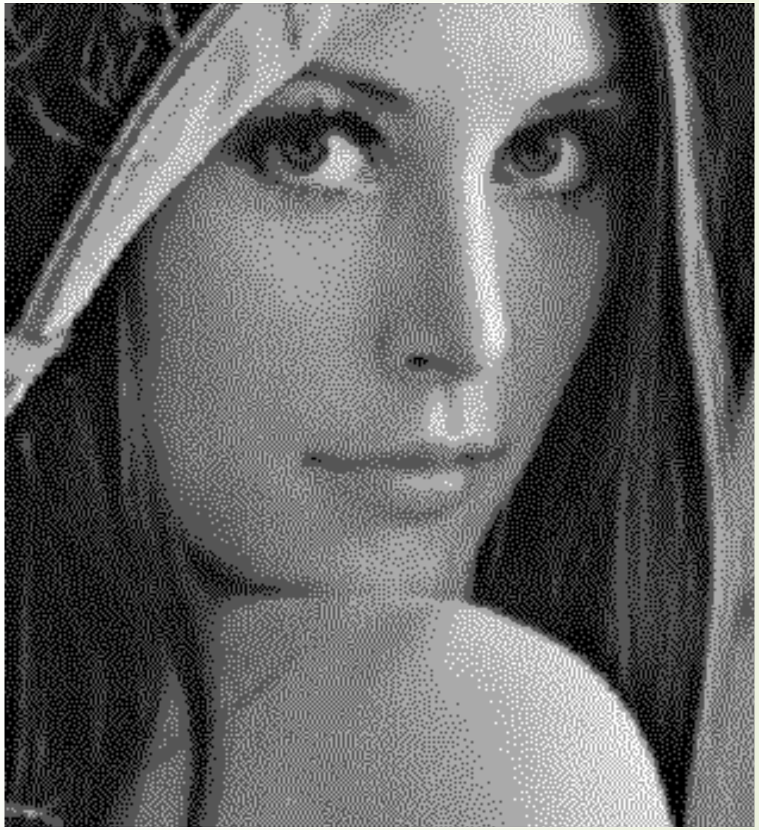
Dithering

- Let see what it happens across two quantization levels on a scanline



Dithering: example

Lena 3 bpp uniform quantization



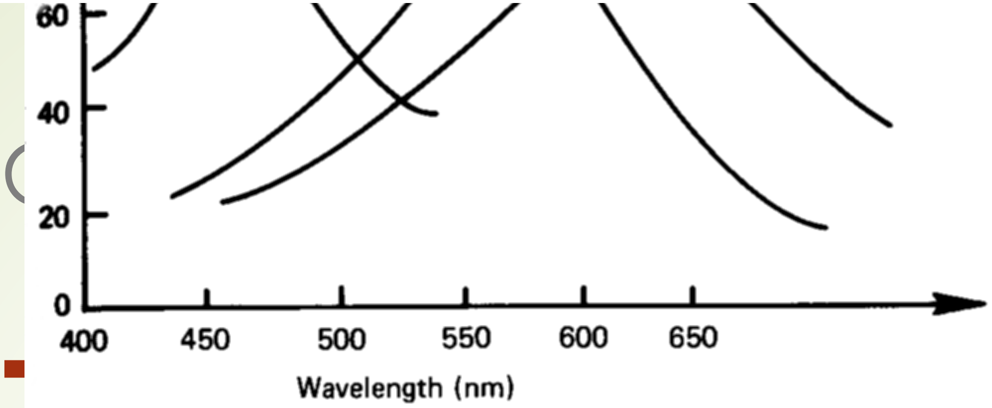
Lena 3 bpp dithering

Halftoning

- …and in the limit case that we have just 1 bpp?
- It is still possible to represent the image with acceptable perceptual quality
  - Again we use dithering, but increasing the resolution and applying 1 bit quantization (black&white)
  - It is possible to use random noise or deterministic patterns
  - It is the process used in printing (e.g., laser printing, offset printing)

Color representation

- A peculiar characteristic of image signals is **color**
  - Color information is not trivial to represent, a single value is no more sufficient, we have to introduce a vector representation
- How do we represent colors? It took a lot of time and effort to answer this question, some basic steps:
  - 1802 **Thomas Young** introduces the **trichromatic theory**: the HVS perceives the color sensation thanks to the combined response of 3 types of receptors (cones)
  - 1850-1920 (Helmholtz, Grassman, Schrodinger): **tristimulus theory**: colors can be generated by appropriate mixtures of 3 primary colors. Primary color sources may change (**metamerism**)
  - 1931 **CIE** (Commission internationale de l'éclairage): the **RGB** and **XYZ** color spaces are defined and standardized



- l receptors:
  - Rods → are sensitive to monochromatic light (b/w)
  - Cones → divided in 3 types, sensitive to different wavelengths (color)

Response of cones

How to generate colors

- Given a color C(λ), the way we perceive it is a function of the 3 cones' responses to the stimulus

$$\alpha_i(C) = \int_{\lambda_{min}}^{\lambda_{max}} S_i(\lambda) C(\lambda) d\lambda \; ; \; i \in [0,1,2]$$

- To synthesize a color that provides the same responses $\alpha_i(C)$ we need 3 sources $P_k(\lambda)$ such that:

$$\alpha_i(C) = \int_{\lambda_{min}}^{\lambda_{max}} S_i(\lambda) \left[\sum_{k=0}^{2} \beta_k P_k(\lambda) \right] d\lambda =$$

$$\sum_{k=0}^{2} \left[\beta_k \int_{\lambda_{min}}^{\lambda_{max}} S_i(\lambda)\, P_k(\lambda) d\lambda \right] = \sum_{k=0}^{2} \beta_k \alpha_{i,k}$$

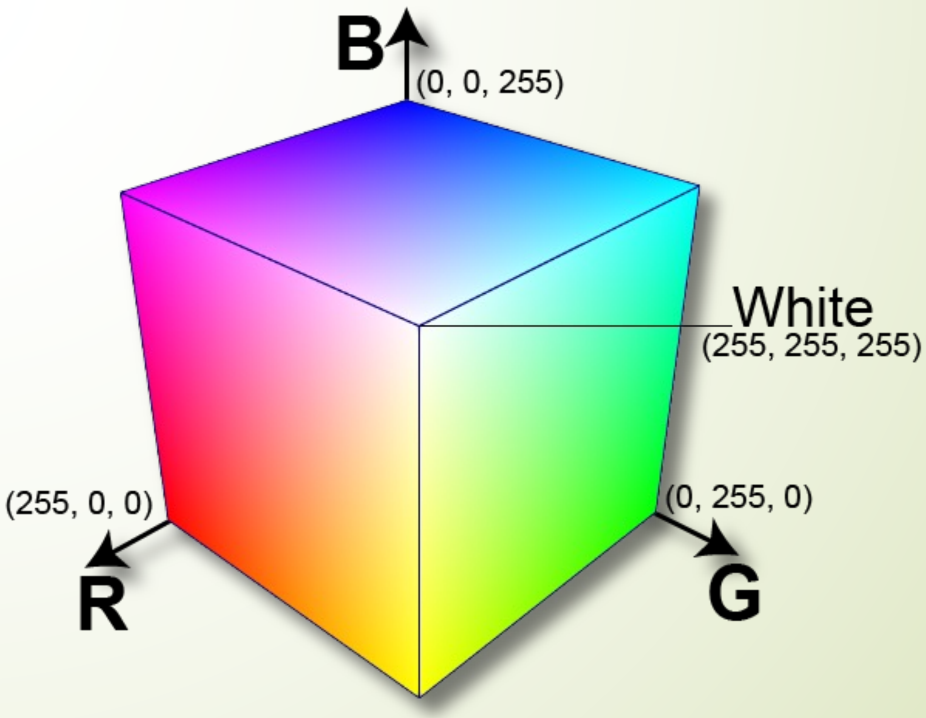
where $\alpha_{i,k}$ is the response of the i-th cone to the k-th source

Color spaces

- We can therefore theoretically define an unlimited number of **color spaces** varying the 3 primary sources P_k
- Not all the spaces will be useful, however
  - Some color space may be physically unfeasible
  - Some color spaces may be incomplete (to generate some colors we need negative components β_k) or overcomplete (some licit mixtures produce colors not perceived by the HVS)
  - Some color spaces are less suitable for a given application (e.g., to measure color distances)

RGB color space

- Red, Green, Blue color space (CIE 1931) can be represented as a cube, where the main axes represent the intensity of the three primary colors (R = 700 nm, G = 546.1 nm, B = 435.8 nm)
  - Along main axes we have intensity variations of pure RGB colors
  - Along main diagonal we have greylevels (black and white at vertices)
  - Opposite vertices represent complementary colors CMY (cyan, yellow, magenta)



Color synthesis

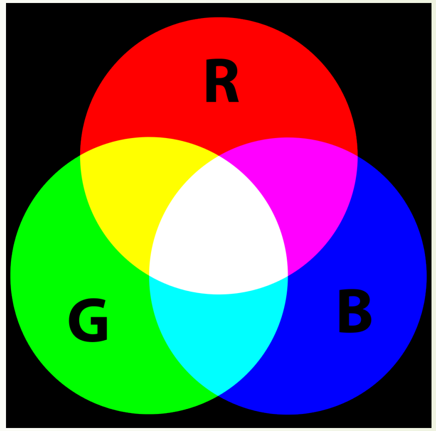
- Based on the RGB color space, we can define 2 physically realizable models for color synthesis

  - **Additive synthesis** of colors (RGB)

    - start from black

    - mix primaries to achieve the desired color

    - typical of TVs, projectors, monitors

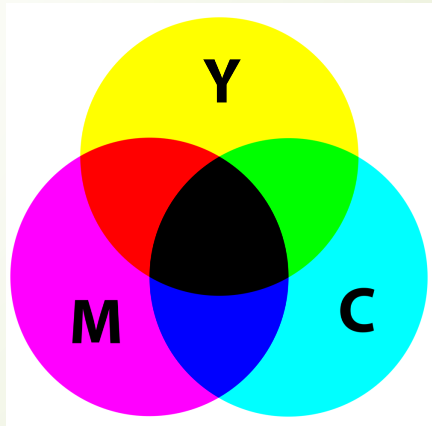


  - **Subtractive synthesis** of colors (CMY)

    - start from white

    - mix primaries to achieve the desired color

    - typical of printers



Color quantization

- Two possibilities:
  - **Quantize colors in 3D space** (vector quantization): perhaps more efficient in terms of bpp but complex
  - **Quantize color components**: simple extension of what we've already seen, straightforwardly applied to color components
- Typically the second solution is preferred
  - Again, to preserve the byte alignment each component is quantized at 8pbb, for a total of 24 bpp
  - This means about **16 millions of colors** (often referred to as **true color**)

NB. This is well beyond the perceptual capabilities of our visual system

Color Conversion

- There are dozens of color spaces tailored to different applications
  - Typically the conversion between color spaces is just a **linear transformation** (a matrix operation)
  - Example1: **RGB to YC_bC_r** space, used in JPEG standard

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

  - Example2: **RGB to YUV** space, used in analog TV transmission (NTSC)

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Digital image filtering

- Now that we finally have a digital image, let see how to extend the concept of **filtering in space and frequency**
- Let start from filtering in space
  - Let i(x,y) be an analog **image** and h(x,y) the PSF of a **linear, space-invariant system**
  - We apply AD conversion to both image and PSF to obtain their digital counterparts i(n,m) and h(n,m)
  - Since h(n,m) has typically infinite extension, we have also to **window it in [K,L]** to obtain the finite-length discrete PSF $h_{DW}(n,m)$
  - Extending the concept already seen for 1D signals, the filtered image will be given by the **2D discrete convolution**:

$$\boxed{i_F(n,m) = i(n,m) * h_{DW}(n,m) = \sum_{k=-K/2}^{K/2} \sum_{l=-L/2}^{L/2} i(n-k,m-l) h_{DW}(k,l)}$$

Discrete 2D convolution

$x(n,m)$

1	2	1	0
0	1	1	1
1	0	2	0

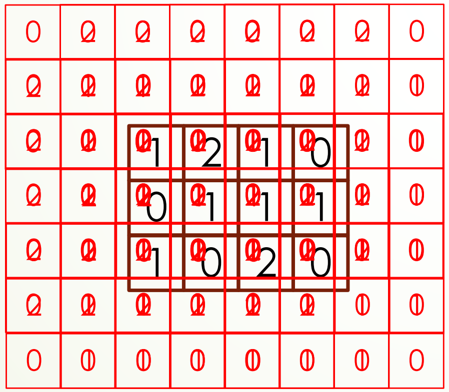
$h_{DW}(n,m)$

0	1	0
1	1	2
0	2	0

$\rightarrow h_{DW}(-n,-m)$

0	2	0
2	1	1
0	1	0

$x(n,m) * h_{DW}(n,m)$



$y(n)$

0	1	2	1	0	0
1	3	6	6	3	0
0	4	6	8	3	2
1	1	6	4	6	0
0	2	0	4	0	0

Discrete 2D convolution: Pseudo-code

```
int input[N][M], output[N][M]         // input and output images
int PSF[K][L]                         // point spread function
int ks = floor(K/2)                   // half h-size of filter
int ls = floor(L/2)                   // half v-size of filter
load(input), load(PSF)                // load image and kernel
for m in ks...N-ks-1 {                // h-scan image w/o borders
  for n in ls...M-ls-1 {              // v-scan image w/o borders
    tmp = 0                           // init var
    for k in -ks...ks {               // h-scan filter
      for l in -ls...ls {             // v-scan filter
        tmp += (input[m+k][n+l]*imp_resp[k+ks][l+ls]) //filter
      }}
    output[m][n] = tmp                // write result to output img
  }}
```

Complexity and separability

- Looking at the pseudo-code, it is easy to see that the 2D discrete convolution has quadratic complexity with respect to the PSF size
- Sometimes, the complexity can be reduced by applying the **separability property**
- A filter is separable if $h(m,n) = h_1(1,m) * h_2(n,1)$, then:

$$i(n,m) * h(m,n) = i(n,m) * h_1(1,m) * h_2(n,1)$$

i.e. a cascade of two 1D convolutions (linear complexity)

- For instance, the following filter is separable:

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

NB. Separability is not granted in general !

DFT 2D

- To see how filtering works in the frequency domain, we have first to extend the frequency representation to discrete 2D domain (DFT 2D)
- Also in this case, the extension is rather straightforward
  - The basis functions become 2D:

$$\boxed{e^{-\frac{j2\pi}{N}kn} \rightarrow e^{-j2\pi\left(\frac{kn}{N}+\frac{lm}{M}\right)}}$$

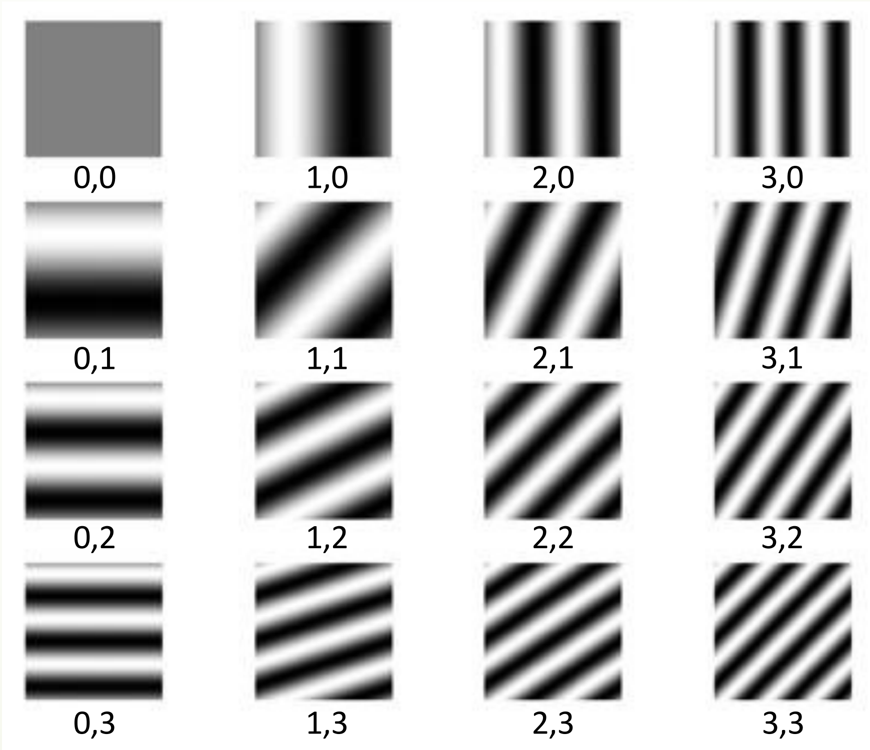
where (n,m) and (k,l) = space and frequency indexes, respectively

NB1. The above basis functions suggest that the 2D frequency representation can be split into ***2 orthogonal components*** *(vertical and horizontal), diagonal components derive from their combination*

NB2. Again, we will get a number of coefficients equal to the number of samples (and a transformed image of the ***same size*** *of the original one)*

DFT basis images

- Since basis functions are 2D we can refer to them as basis images
  - To transform an NxM image, we need NxM basis images
  - Each basis image has a size NxM



Source: Xiaojun Qi, Basic DIP, docplayer.net

DFT-IDFT 2D

- As for the 1D case, we can define the direct and inverse DFT as:

$$I(k,l) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} i(n,m) \cdot e^{-j2\pi\left(\frac{kn}{N}+\frac{lm}{M}\right)} \qquad \text{DFT 2D}$$

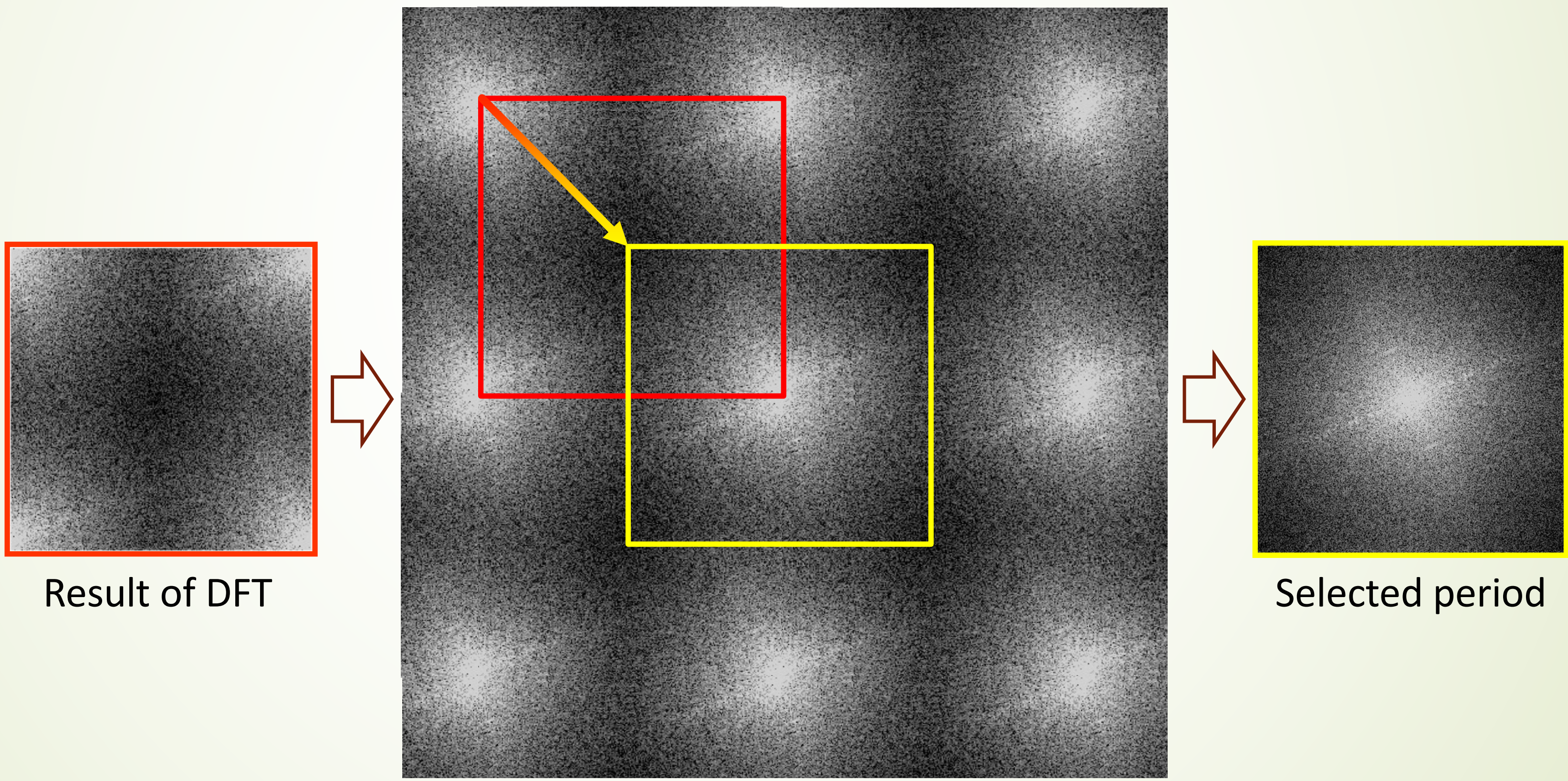
$$i(n,m) = \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} I(k,l) \cdot e^{j2\pi\left(\frac{kn}{N}+\frac{lm}{M}\right)} \qquad \text{IDFT 2D}$$

- 0,0 → continuous wave component (DC-term)
- 0,i → i-th pure vertical frequency (increasing with i)
- j,0 → j-th pure horizontal frequency (increasing with j)
- i,j → mixed frequencies(increasing with i,j)

NB. Also in this case, FFT can be used to speed up the transformation, with complexity O(N logN)

DFT 2D: conventional representation

- The transformed domain is discrete and periodic (sampling)
  - Conventionally, we place the “zero” frequency in the center



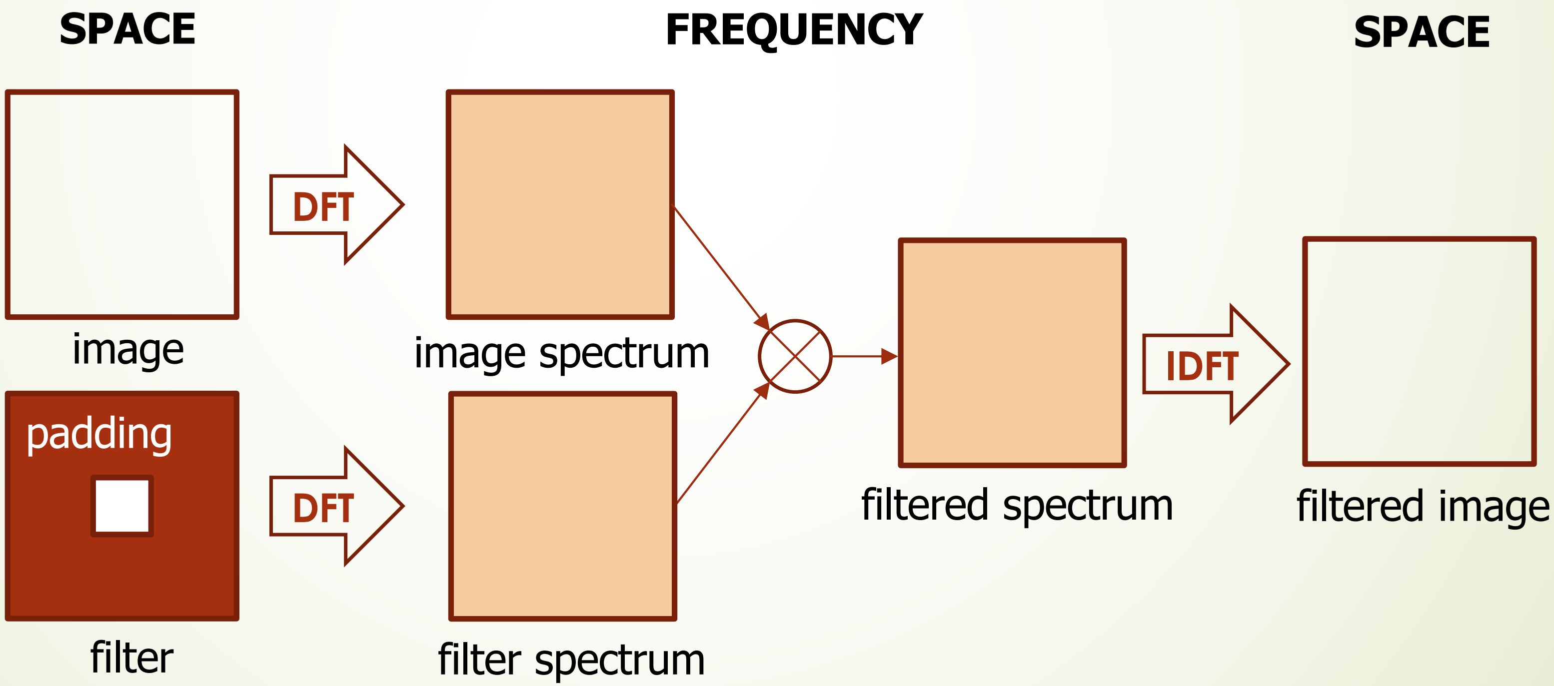
Result of DFT

Selected period

DFT periodic function

Image filtering in DFT domain

- It is a trivial extension of the method we have introduced for 1D discrete signals
  - Apply DFT to 2D image and zero-padded PSF
  - Multiply the frequency spectra and apply IDFT to the result



What we've learned in this section

- Images are signals in multiple dimensions (2 or more)
- Their acquisition relies on sensing (mimicking human senses)
- 1-D signals and systems theory can be straightforwardly extended to m-D (representation in space and frequency, system response)
- Also A/D conversion is readily extended, but visual effects of sampling and quantization need to be addressed appropriately (it's important to understand how HVS works and adapt to it)
- Color is a peculiar feature of images and needs special care. Various representations are available for different scopes.
- Filtering in 2-D discrete domain, both in the spatial and frequency domains, is a direct extension of 1-D case