Image Processing

Riccardo Caprile

March 2022

1 Images

1.1 Image Acquisition

Images cannot exist without light. To produce an image , the scene must be illuminated with one or more light sources.

Certain modalities such as fluorescent microscopy, astronomy , MRI , and X-ray tomography do not fit this model and the image formation is more complex.

Image observed using different bands

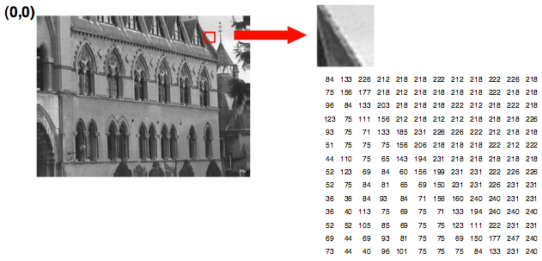
The same scene can be observed acquiring only a specific band of the light. For instance the same area(Washington D.C) produces the following images if observed using different wavelength bands.

Image Formation

- **Geometric parameters** → type of projections, position and orientation of the camera , perspective distortion
- **Optical parameters** → lens type , focal length, field of view, aperture
- **Photometric parameters** → lens type , intensity , direction of illumination; reflectance properties of surfaces , sensors structure.

A monochrome image is an array of values . There are two types of discretization involved :

- Spatial sampling (pixels)
- Intensity quantization (grey level value)



1.2 Digitalization

The digitalization process is formed by two steps :

- **Measurement**
- **Conversion from analog to digital**

In the first step the physical quantity to be represented is measured by an appropriate device that converts it into an electrical continuous signal . In the second step the electrical signal is converted into a digital signal.

CCD/CMOS

Light falling on an imaging sensor is usually picked up by an active sensing area , integrated for the duration of the exposure - usually expressed as the shutter speed in a fraction of second and then passed to a set of sense amplifiers. The two main kinds of sensor used in digital still and video cameras today are charge-coupled device (CCD) and complementary metal oxide on silicon (CMOS).

In a **CCD**, photons are accumulated in each active cell during the exposure time. Then , in a transfer phase , the charges are transferred from cell to cell in a kind of "bucket brigade" until they are deposited at the sense amplifiers, which amplify the signal and pass it to an analog-to-digital converter.

Whereas , a **CMOS** imaging chip is a type of image sensor that has an amplifier for each pixel instead of the few amplifiers of a CCD , this results in less area of the capture of photons than a CCD, but this problem has been overcome with microlens in front of the photodiode that direct the light into the photodiode.

1.3 Quantization

Numer of bits (B)	Depth (2^B) i	Max value	Memory occupacy [512x512] uncompressed image
1	2	1	32 Kb
8 (unsigned char)	256	255	(512x512x8)bit = 256 Kb
16 (int)	65536	65535	(512x512x16)bit = 512 Kb
24 (color: 3 int)	~$16x10^6$	255 (per color)	(512x512x8)x3 bit = 768 Kb
32 (long int,float)	$4x10^9$	2^{32}-1	(512x512x32) bit=1 Mb


1 bit


8 bit (1 byte)


16 bit (2 byte)


24 bit (true color)

1.4 Representation

Pixel Representation

Let us imagine an image to be digitalized is overlaid with a regular grid : This grid is referred to as sampling grid , each element of the grid will contain a portion (region) of the image . The whole portion will be approximated by a unique average value. A coarse sampling grid produces an image with fewer details.

The size of an image

The size of the image is given by the number of pixels composing it. The size is conventionally expressed by the number of rows times the number of columns of the image matrix , e.g 640x480

Resolution

Given an image with a fixed size in pixels , it can be visualized at different sizes(in mm) on various supports (paper , monitor). The visualization is controlled by the resolution. The resolution depends on the size of the image and the size of the support. It is measured in dots/cm or , more frequently , dots/inches(dpi). The resolution is related on how dense are the elements on the support.

Greyscale

In photography and computing , a **grayscale image** is an image in which the value of each pixel is a single sample , that is , it carries only intensity information. Images of this sort , are composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest. **Grayscale images** are distinct from one-bit bi-tonal black and white images , which in the context of computer imaging are images with only two colors, black and white. Grayscale images have many shades of gray between.

Color Spaces

- **RGB** → is short for Red, Green , Blue. It is the color of the light emitted from your computer monitor; when RGB light is combined, the image gets brighter. RGB is an additive color space and is light-based. You add Red,Blue and Green to get white.
- **CMYK** → is short for Cyan, Magenta , Yellow , Black. These arethe inks used in 4-color printing; when the inks are layered on top of each other the image gets darker. CMYK is subtractive , you mix together the colors to get black

1.5 Image Processing

Image Histogram

The histogram of a digital image with intensity levels in the range [0,L-1] is a discrete function :

$$h(r_k) = r_k$$

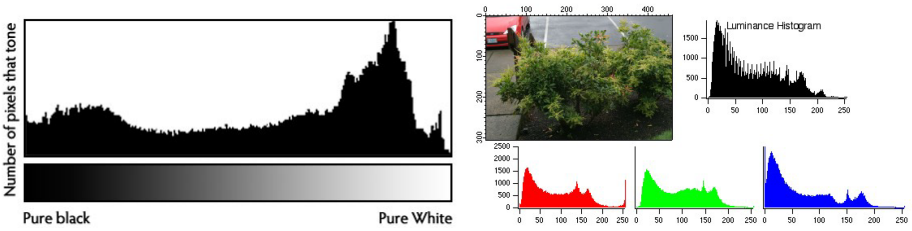
where:

- $\mathbf{r}_k$ is the kth intensity value of the range
- $\mathbf{n}_k$ is the number of pixels in the image with intensity r_k

It is common practice to group similar values while computing the histogram. The range of possible values [0,L-1] can be quantized in bins each of which will group pixels of the image with similar values.

A **histogram** is a graphical representation of the distribution of data. The total area of the histogram is equal to the number of data . The correspondence between histogram and image is not unique:; different images may have the same histogram.

(a) Images with the same histogram

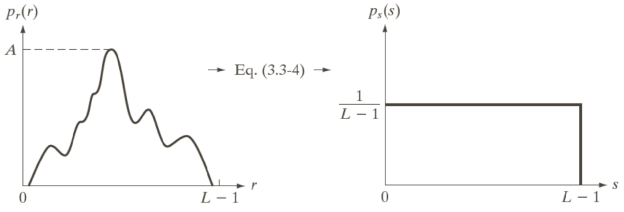


$$p(r_k) = n_k/N\ M \text{ (N x M is the image size)}$$

It can be seen as an estimate of the intensity probability distribution; the area of the normalized histogram is equals to 1. **Histogram processing algorithms** produce transformations on images through their histograms.

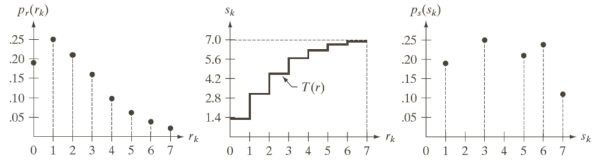
$$s = T(r)\ 0 <= r <= L\text{-}1$$

Histogram equalization Equalization usually increases the global contrast an image. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast.



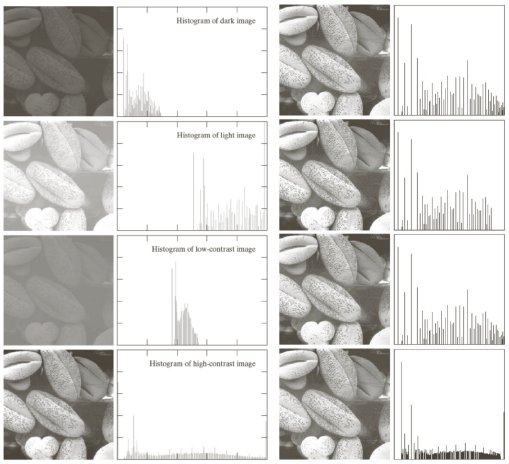
a b

FIGURE 3.18 (a) An arbitrary PDF. (b) Result of applying the transformation in Eq. (3.3-4) to all intensity levels, r. The resulting intensities, s, have a uniform PDF, independently of the form of the PDF of the r's.



a b c

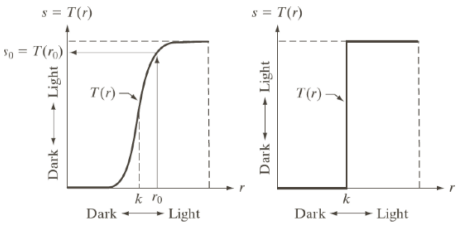
FIGURE 3.19 Illustration of histogram equalization of a 3-bit (8 intensity levels) image. (a) Original histogram. (b) Transformation function. (c) Equalized histogram.

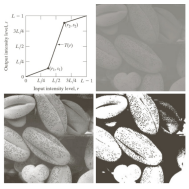


Point Operators

The simplest kind of image processing transforms are point operators , where each output pixel value depends on only the corresponding input pixel value. Examples of such operators include brightness and contrast adjustments as well as color correction and transformations.

Intensity transformation are described by a function that define, for each level of the pixel , the new level. In the following two examples for (a) Contrast stretching and (b) Thresholding





Top-right: Original Image, bottom-left: Contrast stretched image , bottom-right: Threshold image

Neighbourhood operators

In this class of operators an output pixel is obtained starting from a set of neighbouring pixels in the input image. The neighbourhood is usually squared (W x W pixels) and its size may vary W = 3,5,11 ... These operators are often referred as filters.

1.6 Geometric Transformations

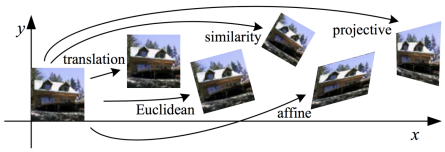
Such transformations modify the position of pixels instead than their value : **transformation of coordinates**. Given a pixel of image I at coordinates p =

(p_1 , p_2) the effect of a transformation H is to move I(p) to q = (q_1 , q_2) that is :

$$q = H(p)$$

and the corresponding image transformation from the input image I to the output image J is

$$J(q) = J(H(p) = I(p)$$



Transformation	Matrix	Preserves	Icon
translation	$\left[\ \boldsymbol{I} \mid \boldsymbol{t}\ \right]_{2\times3}$	orientation	
rigid (Euclidean)	$\left[\ \boldsymbol{R} \mid \boldsymbol{t}\ \right]_{2\times3}$	lengths	
similarity	$\left[\ s\boldsymbol{R} \mid \boldsymbol{t}\ \right]_{2\times3}$	angles	
affine	$\left[\ \boldsymbol{A}\ \right]_{2\times3}$	parallelism	
projective	$\left[\ \tilde{\boldsymbol{H}}\ \right]_{3\times3}$	straight lines	

Translation t = (t_1,t_2)

$$q_1 = p_1 + t_1$$

$$q_2 = p_2 + t_2$$

Rotation of an angle θ (around the origin)

$$q_1 = p_1 \cos\theta + p_2 \sin\theta$$

$$q_2 = -p_1 \sin\theta + p_2 \cos\theta$$

Scaling :

$$q_1 = c \cdot p_1$$

$$q_2 = d \cdot p_2$$

In digital image processing they consist of two steps : 1) A spatial transformation of coordinates according to T , 2) An intensity interpolation to assign intensity values to the spatially transformed pixels on the discrete grid.

2 2D Fourier Transform for Images

2.1 2D Fourier Transform

The 2D Fourier Transform can be easily derived from the 1D

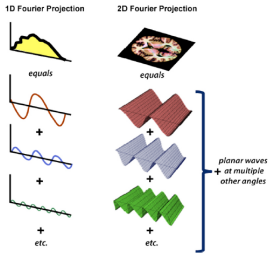
- **Fourier Transform:**

$$F(u,v) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x,y) e^{-i2\pi(ux+vy)} dxdy$$

- **Inverse of the Fourier Transform:**

$$f(x,y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} F(u,v) e^{i2\pi(ux+vy)} dudv$$

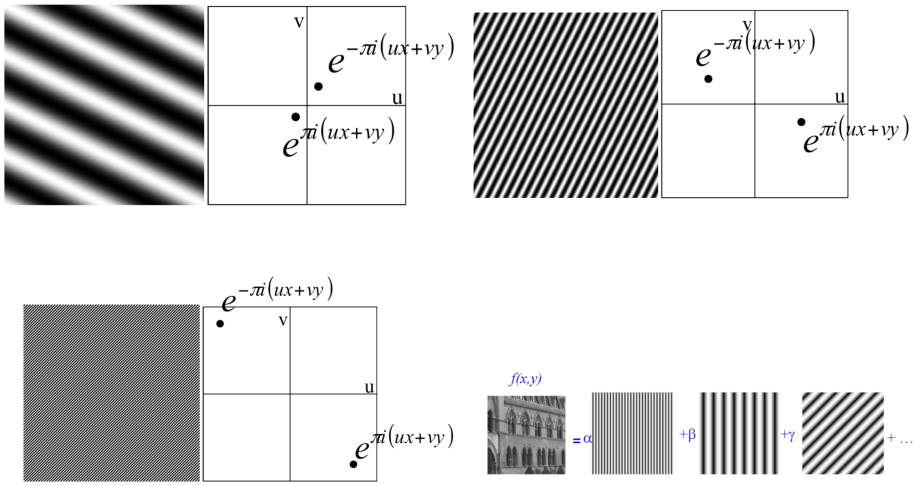
u and v are spatial frequencies.
In general, $F(u,v)$ is complex $F(u,v) = F_R(u,v) + iF_I(u,v)$.



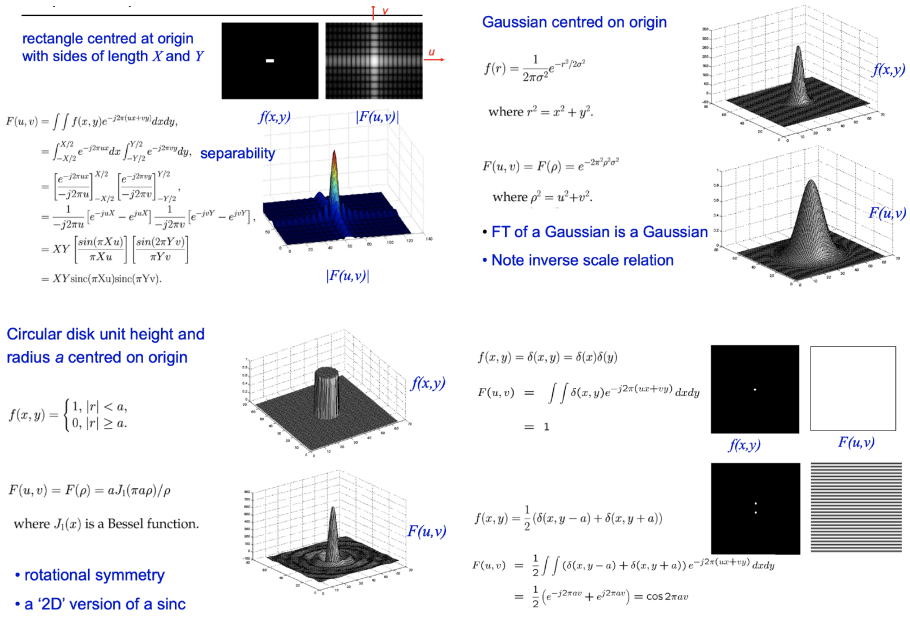
- Similarly to the 1D case, the FT is based on a decomposition into sinusoidal functions that form an orthonormal basis

$$e^{i2\pi(ux+vy)} = \cos 2\pi(ux+vy) + i \sin 2\pi(ux+vy)$$

- the terms are sinusoids on the x, y plane



Fourier Transform pair examples :



Translation and Rotation

The spectrum is insensitive to image translation , but it rotates by the same angle of a rotated image. It is possible to see the differences in the spectrum between the original image and the rotated one.

2.2 Discrete Fourier Transform

DFT of Images In the discrete world , let consider an image as a matrix of size M x N where any element represents the value of the pixel f[m,n] and m = 0, ... , M - 1 and n = 0, ... , N - 1. In order to apply DFT , we must consider the image as a part of its periodic expansion.

$$f[m +- M,n] = f[m,n+- N] = f[m +- M, n +- N] = f[m,n]$$

Then, the Discrete Fourier Transform and its inverse are :

$$F[k,l] = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)e^{-i2\pi(\frac{mk}{M}+\frac{nl}{N})}$$

$$f[m,n] = \frac{1}{MN}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1} F(k,l)e^{i2\pi(\frac{mk}{M}+\frac{nl}{N})}$$

2.3 DFT of an image : useful properties

Complex Conjugate symmetry

An image is usually a 2D real-valued signal. If [m,n] is real , f[m,n] = f*[m,n], then F*[k,l] = F[-k,-l].

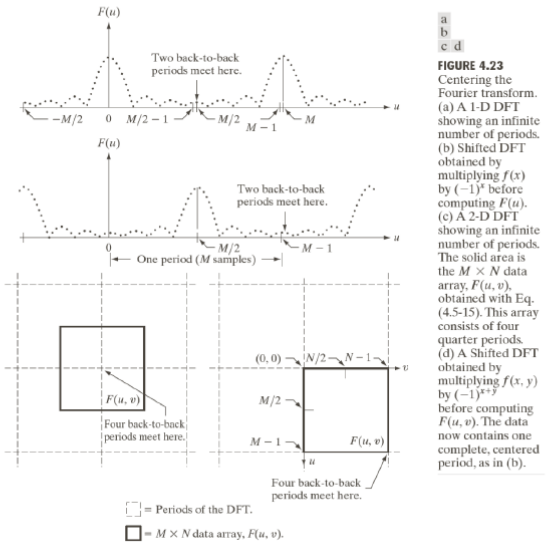
It follows that —F(k,l)— = —F(-k,-l)—, which says that the spectrum of the Fourier transform is symmetric. In other words, there exist negative frequencies which are mirror images of the corresponding positive frequencies.

Periodicity

The spectrum repeats itself endlessly in both directions with period N and M : F [k,l] = F[k + M , l] = F[k,l + N] = F[k + M , I + N]. The N x M block of the Fourier coefficients F[k,l] computed from an N x M image is a single period from this infinite sequence.

2.4 Properties of DFT-2D : shift in frequency

Properties of DFT-2D: shift in frequency



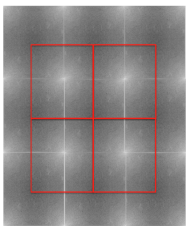
a
b
c d

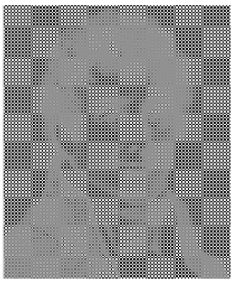
FIGURE 4.23 Centering the Fourier transform. (a) A 1-D DFT showing an infinite number of periods. (b) Shifted DFT obtained by multiplying $f(x)$ by $(-1)^x$ before computing $F(u)$. (c) A 2-D DFT showing an infinite number of periods. The solid area is the $M \times N$ data array, $F(u, v)$, obtained with Eq. (4.5-15). This array consists of four quarter periods. (d) A Shifted DFT obtained by multiplying $f(x, y)$ by $(-1)^{x+y}$ before computing $F(u, v)$. The data now contains one complete, centered period, as in (b).

Reconstructed image (we need to be careful and shift back before reconstruction)

$f_{\{M/2,N/2\}}[k,l] = (-1)^{k+l} f[k,l]$

$F(k - M/2, l - N/2)$

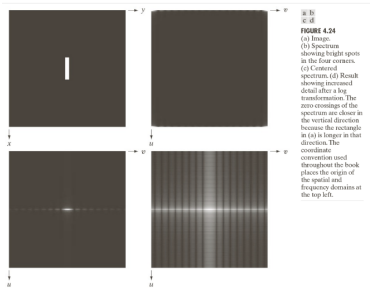




2.5 Visualizing the DFT

An appropriate frequency shift of (M/2,N/2) will take the DC component F[0,0] at the center of the map (F(0,0) = $\sum_{m=0}^{N-1} p_i \sum_{n=0}^{N-1}$ f(m,n) is the image average intensity level).

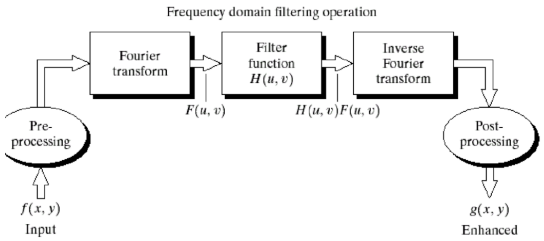
Because the lower frequency amplitudes mostly dominate over the mid-range and high-frequency ones , the fine structure of the amplitude spectrum can be perceived only after a non-linear mapping to the greyscale range [0,255]



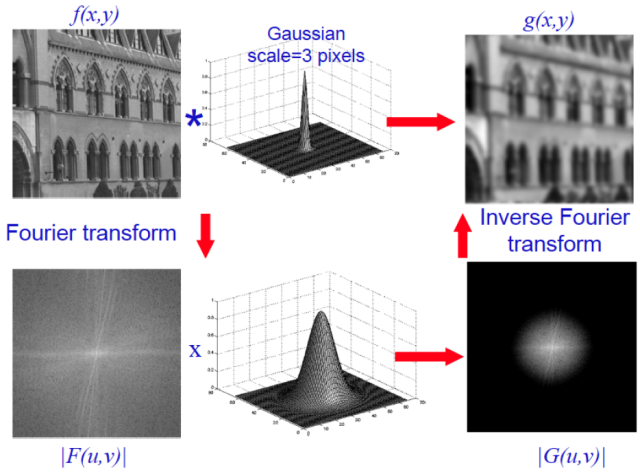
2.6 Fourier Filtering

To filter an image in the frequency :

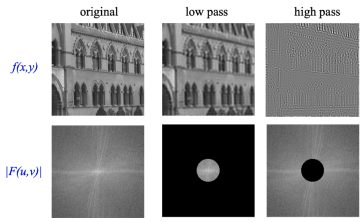
- Compute the DFT of the image F(u,v)
- Select and appropriate filter H (or filter transfer function)
- Multiply F(u,v) by a filter function H(u,v) : G(u,v) = H(u,v) F(u,v)
- Compute the inverse DFT of G(u,v)



2.7 Image filtering and convolution theorem



2.8 Filters in space and frequencies

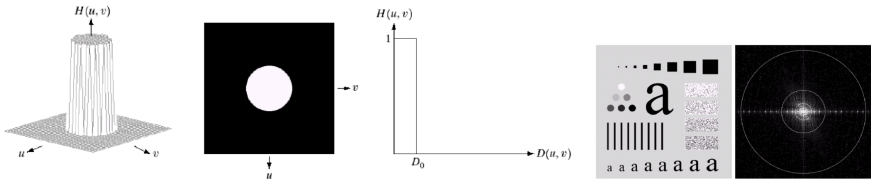


2.9 Image smoothing

Smoothing is achieved in the frequency domain by dropping out the high frequency components. This effect is obtained by applying a low-pass filter. We can take into account different low-pass filters , for instance : Ideal low-pass, Butterworth , Gaussian

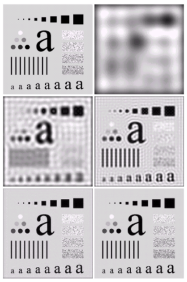
2.9.1 Ideal low-pass filter

Simply cuts off all high frequency components that are a specified D0 from the origin transform.



Above we show an image , its Fourier spectrum and a series of ideal low pass filters of radius 5 , 15 , 30 , 80 and 230 superimposed on top of it.

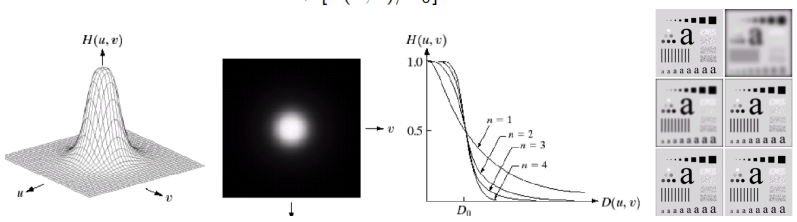
The original image and the five different filtered images D0 = 5,15,30,80,230.



2.9.2 Butterworth low-pass filter

The transfer function of a Butterworth lowpass filter of order n with cutoff frequency at distance D0 from the origin is defined as :

$$H(u,v) = \frac{1}{1 + [D(u,v)/D_0]^{2n}}$$

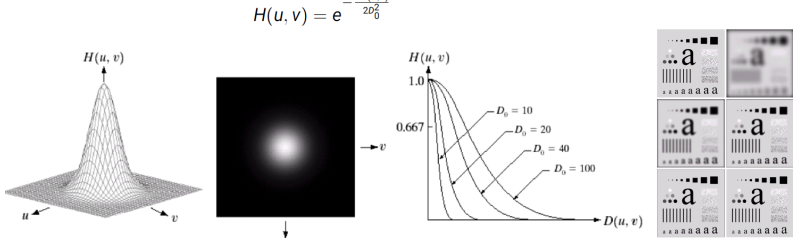


The original image and the five different images filtered by a Butterworth filter of order and D0 = 5,15,30,80,230

2.9.3 Gaussian low-pass filter

The transfer function of a Gaussian lowpass filter with cutoff frequency at distance D0 from the origin is defined as :

$$H(u,v) = e^{-\frac{D^2(u,v)}{2D_0^2}}$$



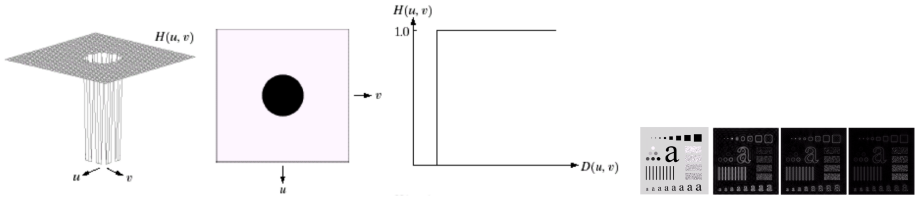
2.9.4 Low-pass filtering comparison

The test image filtered by : (left) ideal low-pass filter D0 = 15 , (center) butterworth low-pass filter D0 = 15 , (right) Gaussian low-pass filter D0 = 15

2.10 Image enhancement

Fine details in images are associated with high frequency components. High pass filters only pass the high frequencies , dropping the low ones. High pass filters are the reverse of low pass filters , so : H_hp(u,v) = 1 - H_lp (u,v)

2.11 Ideal High-pass filter

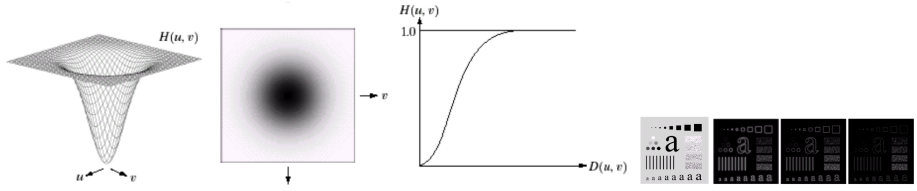


The original image and three different images filtered by an ideal high-pass filter with D0 = 15,30,80

2.11.1 Gaussian high-pass filter

The Gaussian high-pass filter with cutoff frequency at distance D0 from the origin is defined as :

$$H(u,v) = 1 - e^{-\frac{D^2(u,v)}{2D_0^2}}$$



The original image and three different images filtered by a Gauss filter with D0 = 15,30,80

3 Spatial Filters

3.1 Preliminaries : the mathematics of spatial filtering

3.1.1 Convolution

Convolution is defined as the integral of the product of the two functions after one is reversed and shifted :

$$
\begin{aligned}
(f * h)(t) &= \int_{-\infty}^{+\infty} f(\tau) h(t-\tau) d\tau \\
&= \int_{-\infty}^{+\infty} f(t-\tau) h(\tau) d\tau \quad \text{(by commutativity)}
\end{aligned}
$$

Let us consider $g(t) = (f * h)(t)$:

$$
\begin{aligned}
G(\omega) &= \int_{-\infty}^{+\infty} h(t) e^{-i2\pi\omega t} dt = \int_{-\infty}^{+\infty} \left(\int_{-\infty}^{+\infty} f(\tau) h(t-\tau) d\tau \right) e^{-i2\pi\omega t} dt \\
&= \int_{-\infty}^{+\infty} f(\tau) \left(\int_{-\infty}^{+\infty} h(t'') e^{-i2\pi\omega t''} dt'' \right) e^{-i2\pi\omega\tau} d\tau \qquad [t'' = t - \tau] \\
&= F(\omega) H(\omega)
\end{aligned}
$$

And viceversa: $g(t) = f(t)h(t) \iff G(\omega) = F(\omega) * H(\omega)$

3.1.2 Convolution theorem

3.1.3 Discrete convolution

If f and h are defined on integer values : (f * h)[n] = $\sum_{m=-inf}^{+inf} f[m]h[n-m]$ with a finite support (f*h)[n] = $\sum_{m=0}^{M} f[m]h[n-m]$

3.1.4 2D Discrete convolution and filtering

We consider an image f and a filter or kernel k. We obtain g , the filtered version of f.

g[x,y] = (f * k)[x,y] = $\sum_{m,l} f[x-m, y-l]k[m,l]$

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

$f(x,y)$

*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

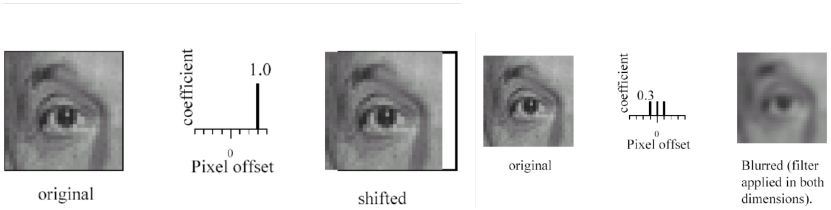
$h(x,y)$

=

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

$g(x,y)$

3.1.5 Filtering Examples



3.2 Smoothing filters

3.2.1 Noise

We only briefly mention the fact real images are affected by different sources of noise. An empirical evidence :

3.2.2 Noise models

Different models of noise can be found in the literature. Pictorial images are often assumed to be affected by some amount of additive noise .

f_r(x,y) = f_i(x,y) + η(x,y) where :

- f_i is the ideal (unknown) image
- f_r is the real (observed) image
- η. is the noise term

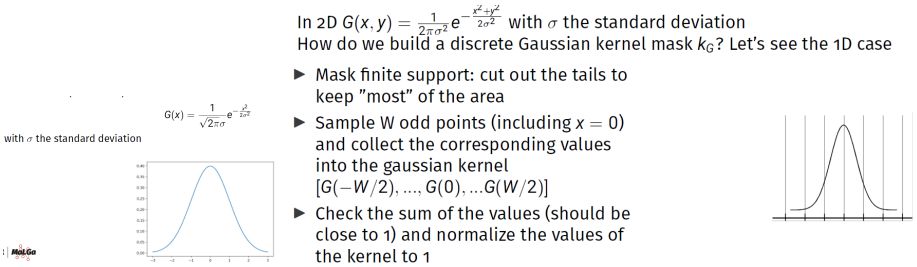
A rather common and effective model for noise is the **Gaussian distribution**

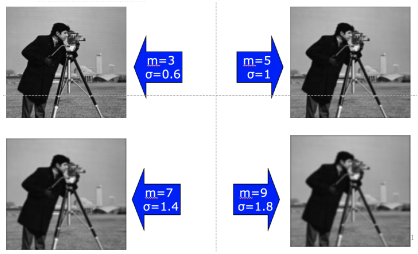
3.2.3 Noise reduction : average filters

With an average filter we replace each pixel by the average of its neighbors and itself. This has the effect of eliminating pixel values which are unrepresentative of their surroundings. This assumes that neighboring pixels are similar and the noise to be independent from pixel to pixel. Average can be represented by an appropriate kernel

3.2.4 Noise reduction : Gaussian filter

The Gaussian smoothing operator is a 2D convolution operator to "blur" images and remove detail and noise. In this sense it is similar to the mean filter , but it uses a different kernel that represents the shape of a Gaussian hump. The Gaussian distribution in 1D has the form :





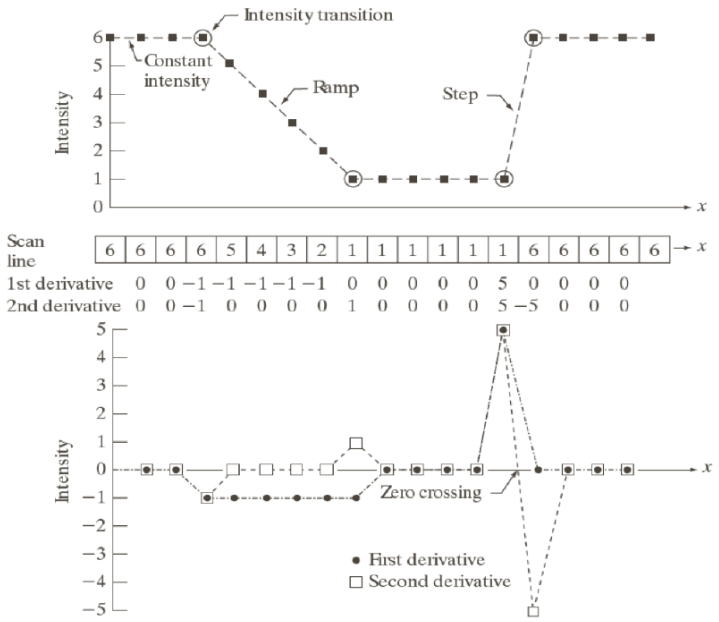
3.3 A parenthesis on efficiency : separable kernels

A single 2D convolution costs O(k^2) is K x K is the size of the kernel mask. Two consecutive 1D filtering operations may be more efficient O(2K) Separable filter k = vv^T are more efficient. If the filter is separable , instead than one 2D convolution we obtain the same effect with 2 consecutive 1D convolutions (f_r = f * v , g = f_r * h) The Gaussian filter and the average filters are separable.

3.4 Enhancement filters

3.4.1 Computing image derivatives

Enhancement filters highlight transitions in intensity. We will use them to estimate finite differences. We will need to cope the fact differentiation enhances interesting discontinuities but also noise.



1st derivative : Zero in constant areas , not zero on ramps.

2nd derivative : Zero in constant areas , Zero on ramps with constant slope, not zero at the beginning and the end of the ramps.

$\frac{\partial f}{\partial x}$ = f(x+1) - f(x)

$\frac{\partial^2 f}{\partial x^2}$ = f(x+1) +f(x-1) - 2f(x)

First derivatives

It may be useful to compute

- Image gradient $\nabla f = grad(f) = [g_x, g_y] = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$
- Magnitude of the gradient $M(x, y) = \sqrt{g_x^2 + g_y^2}$

  Notice: it's not linear (we cannot implement it with a kernel)

Examples of 1st derivative filters:

K | -0.5 | 0 | 0.5 — Central difference

K | -1 | 1 — Forward difference

$$g_x = K * f$$
$$g_y = K^\top * f$$

Rx | -1 | 0 / 0 | 1

Ry | 0 | -1 / 1 | 0

Roberts filter

$$g_x = R_x * f$$
$$g_y = R_y * f$$

Second derivatives

- Zero-crossings are good indicators of sharp variations
- The Laplacian is a 2D isotropic measure of the 2nd spatial derivative of an image

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

- The Laplacian, as a sum of derivatives (which are linear) is linear and can be represented through an appropriate kernel

0	1	0
1	-4	1
0	1	0

Rotation invariant for 90 deg increments

1	1	1
1	-8	1
1	1	1

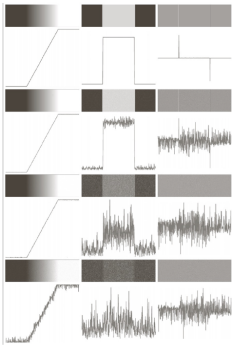
Rotation invariant for 45 deg increments

3.4.2 What do we do with derivatives

As we will see later in the course derivatives allow us to highlight signal discontinuities. **Gradient** : edge detection , corners , image sharpening **Laplacian** : edge detection , blob-=like features , image sharpening.

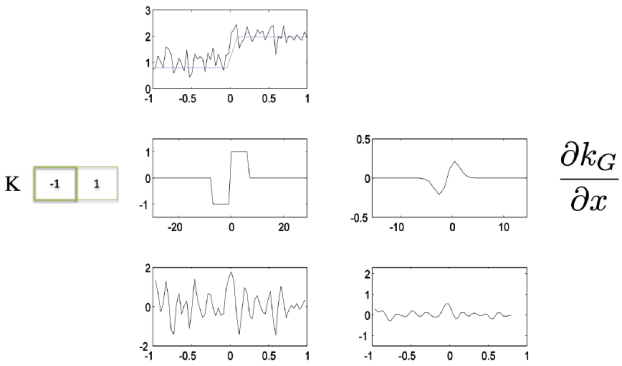
3.4.3 Derivatives and noise

So far, we discussed the ideal noise-free case. In the presence of noise , enhancement filters also enhance noise. General scheme : 1. Smoothing the image with a low pass filter , 2. Computing the derivative

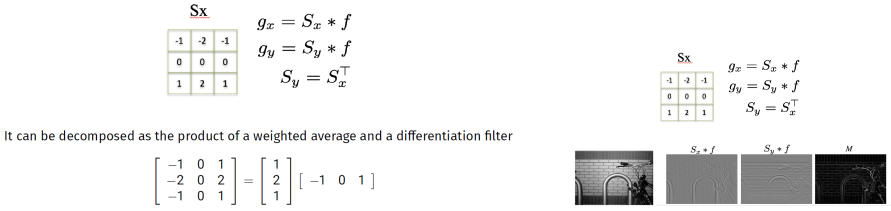


From the properties of convolution :
$\frac{\partial^n}{\partial u}(k * f) = \frac{\partial^n k}{\partial u} * f$
Therefore the previous scheme is equivalent to : 1. Computing the derivative filter, 2. Smooth the image with the derivative of the filter. This alternative has some advantage : Since the kernels are smaller than the image , we will have fewer computations and often the derivative of kernels can be pre-calculated and this would give us only once convolution

3.4.4 Derivative of the Gaussian



3.4.5 Example : the Sobel operator



3.4.6 Non linear filter

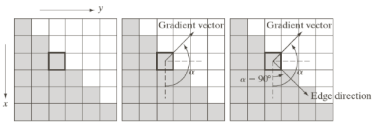
In contrast to frequency filtering , in the space domain we could design non linear filters. A well-known example is the median filter , used to contrast the salt-and-pepper noise. Another interesting non linear filter is the bilateral filter.

4 Feature Detection

4.1 Edge Detection

4.1.1 Edges: definition

- Edge position
- Edge normal
- Edge strength



- We refer mainly to step edges
- The *ideal* step edge models the type of structure we are interested in

$$I(x) = \begin{cases} 0 & x < 0 \\ A & x \geq 0 \end{cases}$$

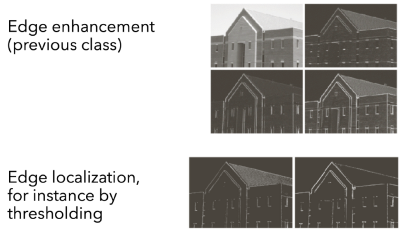
4.1.2 Criteria for optimal edge detection

Good detection : The optimal detector must minimise the probability of false positives as well as that of missing real edges?

Good Localization. The edges must be as close as possible to the true edges?

Single response constraint : The detector must return one point only for each true edge point

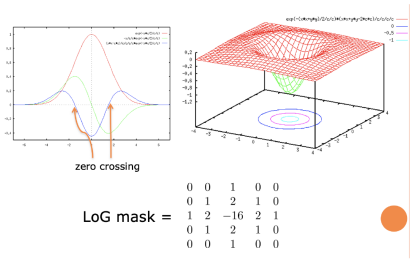
4.1.3 Edge detection by thresholding : Sobel Algorithm



4.1.4 Edge detection by zero crossing

The simplest way is to compute the gradient magnitude and then threshold it. A more accurate way to compute the gradient maxima is to look for **zero-crossing** of the second derivative. This is the **Laplacian of Gaussian (LoG**

$$\nabla \cdot J_\sigma(\mathbf{x}) = [\nabla^2 G_\sigma](\mathbf{x}) * I(\mathbf{x})$$



- Filter the image with a LoG mask
- Locate edge elements then the sign between adjacent elements changes.
- This approach produces edge chains in closed loops.
- It is quite common to use a threshold.

4.1.5 Approximating the LoG

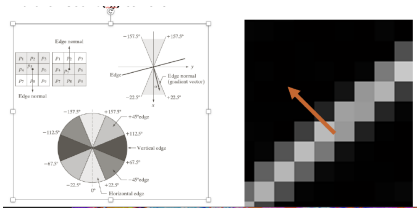
It is common practice to approximate a LoG filter with a Difference of Gaussians (DoG) of different amplitude. This is particularly useful in multi-scale models.

4.1.6 Edge detection : Canny Algorithm

The Canny edge detector is still one of the most largely used algorithms. It implements an approximation of the optimal step edge detector. The algorithm :

1. Edge enhancement
2. Non-maxima suppression
3. Hysteresis thresholding

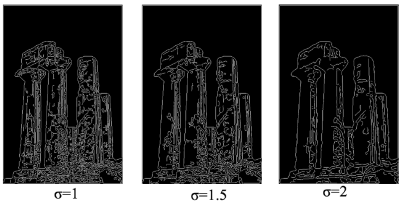
We compute the image gradient J taking into account the presence of noise.
We compute the edge intensity : E_s(i,j) = $\sqrt{J^2{}_x(i,j) + J^2{}_y(i,j)}$
and estimate the edge normal : $E_0(i,j) = \arctan \frac{J_y}{J_x}$
For each pixel (i,j) given a sampling of directions (D = 0,45,90,135).
Look for direction d in D that best approximates E_o(i,j). If E_s(i,j) is smaller than its neighbours in the direction d then set E_s(i,j) to zero.



Given two thresholds t1 ¿ t2. For each pixel (i,j) if E_s(i,j) ¿ t1 : starting from E_s (i,j) follow the chains of connected local maxima in both directions perpendicular to the edge normal as long as E_s (k,h) ¿ t2. Mark all visited points and save a list of the locations of all points in the connected contour found.

The output is a set of edge chains.

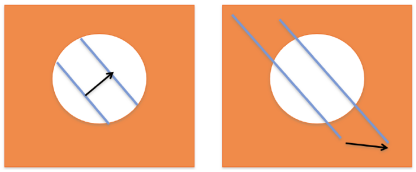
4.1.7 Canny Algorithm : the role of σ



4.2 Corner detection

4.2.1 Good features

Patches with large contrast changes are easier to localize. Straight line segments with a single orientation suffer the **aperture problem**



We observe how patches with gradients in at least two (significantly) different orientations are the easier to localize.

This can be formalized by analysing a simple matching criterium.

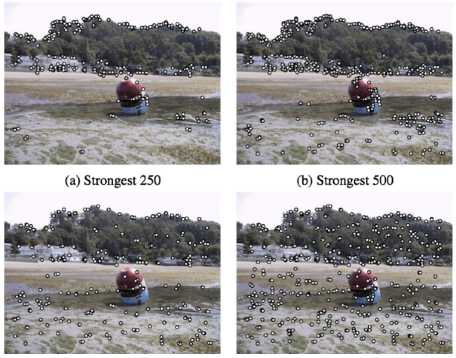
In particular we use it to check how stable the patch is wrt small variations in position u (SSD autocorrelation function):

$E_A C(\mathrm{u}) = \sum_i [I(x_i + u) - I(x_i)]^2$

4.2.2 Corner detection : algorithm sketch

Given an image point q we consider a neighborhood Nq and compute its auto-correlation matrix

$$A_q = \begin{bmatrix} \sum_{p \in N_q} I_x^2(p) & \sum_{p \in N_q} I_x(p) I_y(p) \\ \sum_{p \in N_q} I_x(p) I_y(p) & \sum_{p \in N_q} I_y^2(p) \end{bmatrix}$$



(a) Strongest 250 (b) Strongest 500

- We diagonalise it

$$\mathrm{eig}(A) = [\lambda_1, \lambda_0] \quad \lambda_1 > \lambda_0$$

$$A = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_0 \end{bmatrix}$$