Representation of Images and Videos data

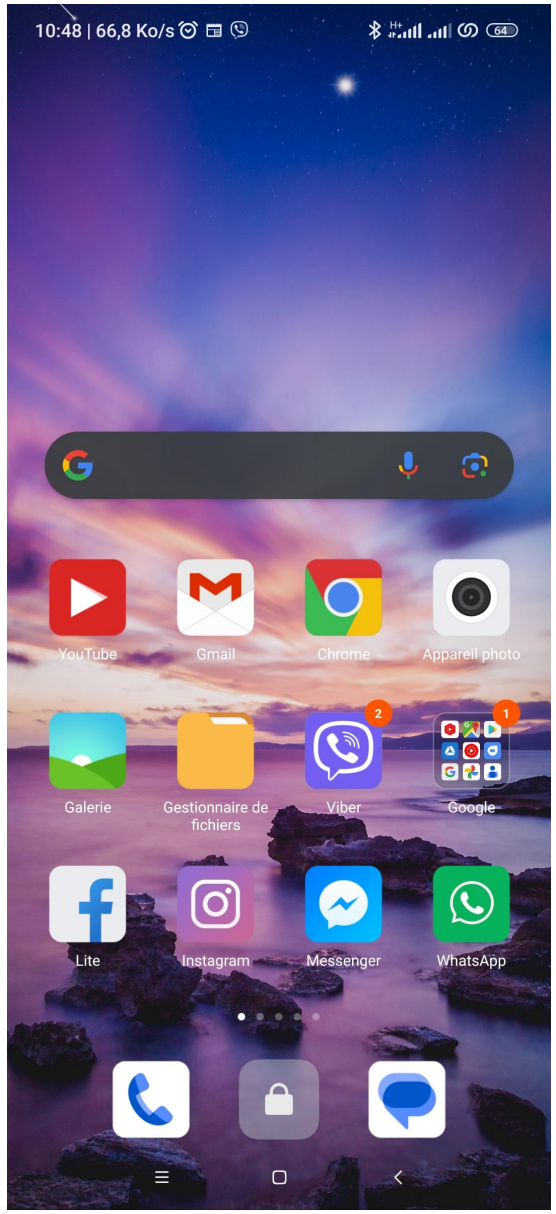
Faculty of Electrical Engineering, USTHB [akourgli@usthb.dz
http://perso.usthb.dz/~akourgli/

Introduction

Images and videos are everywhere!!!

- Personal Photos & videos
- Social networks: Insta, FB, discord, emoji, youtube , tiktok
- Messaging and video call platform: whatsapp , messenger , viber , skype
- Video-conferencing platform: meet , zoom, teams, etc.
- Navigation: maps

Communication support like language



Faculty of Electrical Engineering, USTHB [akourgli@usthb.dz
http://perso.usthb.dz/~akourgli/

Introduction

Endless applications!!!

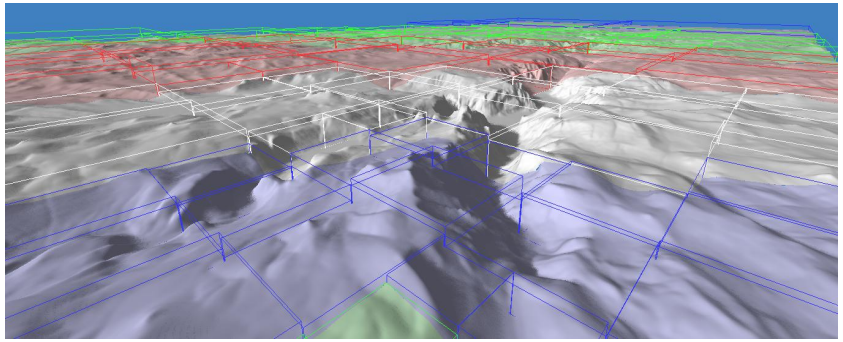
Security

Virtual reality

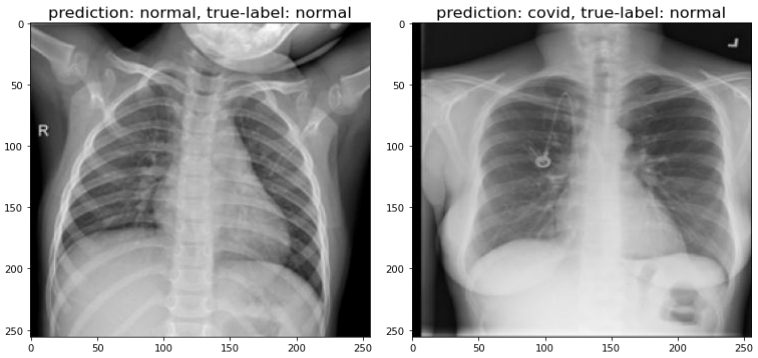
Robotics

Industry

Automatic guidance and tracking

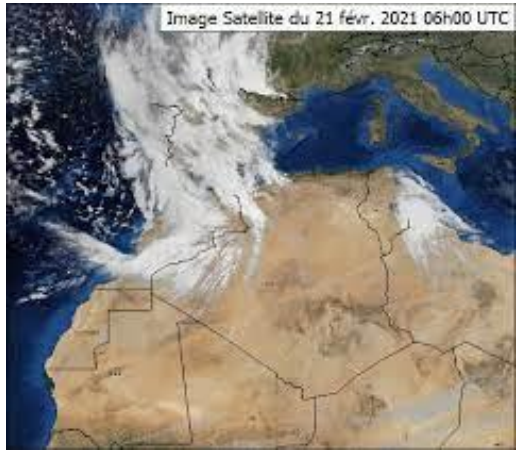


Medicine (Cytology, Tomography, Ultrasound, etc.

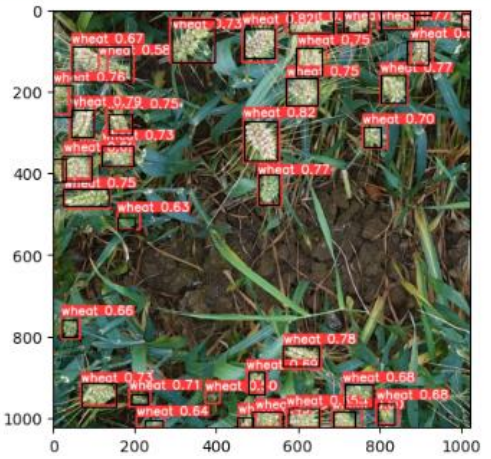


Biometrics

Aerial and space imagery



Agriculture



Faculty of Electrical Engineering, USTHB [akourgli@usthb.dz
http://perso.usthb.dz/~akourgli/

Introduction

Video uses

Learn

Bébé Requin Danse | Chante et danse! les enfants

14 Md de vues • il y a 7 ans

Pinkfong Baby Shark - Kids' Songs & Stories

PINKFONG! non. 1 app enfants choisi par 100 millions d'e

مطبخ ام وليد كيكة الكرامال و القهوة

7,5 M de vues • il y a 7 ans

Oum Walid

بحلية بالكرمال و القهوة

How To Fix a Water Damaged Laptop

61 M de vues • il y a 3 ans

HowToBasic

Today I show you how to fix a water damaged laptop. Acc

Learn Python - Full Course for Beginners [Tutorial]

43 M de vues • il y a 5 ans

freeCodeCamp.org

This course will give you a full introduction into all of the core concepts in python

Sous-titres

Introduction | Installing Python & PyCharm | Setup & Hello World | D

HAVE BEEN / HAS BEEN / HAD BEEN - Examples

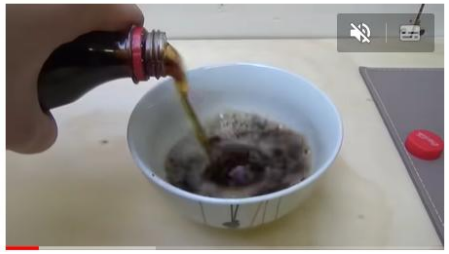
39 k vues • il y a 21 heures

English with Lucy

How do we use HAVE BEEN, HAS BEEN and HAD BEEN? L

Nouveau 4K

Introduction | Subject-Verb Agreement | Posi



How To Fix a Cracked iPhone Screen

60 M de vues • il y a 8 ans

HowToBasic

Today I show you how to easily fix any cracked smartphor

Abdos en 2 SEMAINES | Get Abs in 2 WEEKS |

531 M de vues • il y a 4 ans

Chloe Ting

Abdos ! Abdos ! Abdos ! Tout le monde m'a demandé un calendrier cc

Sous-titres

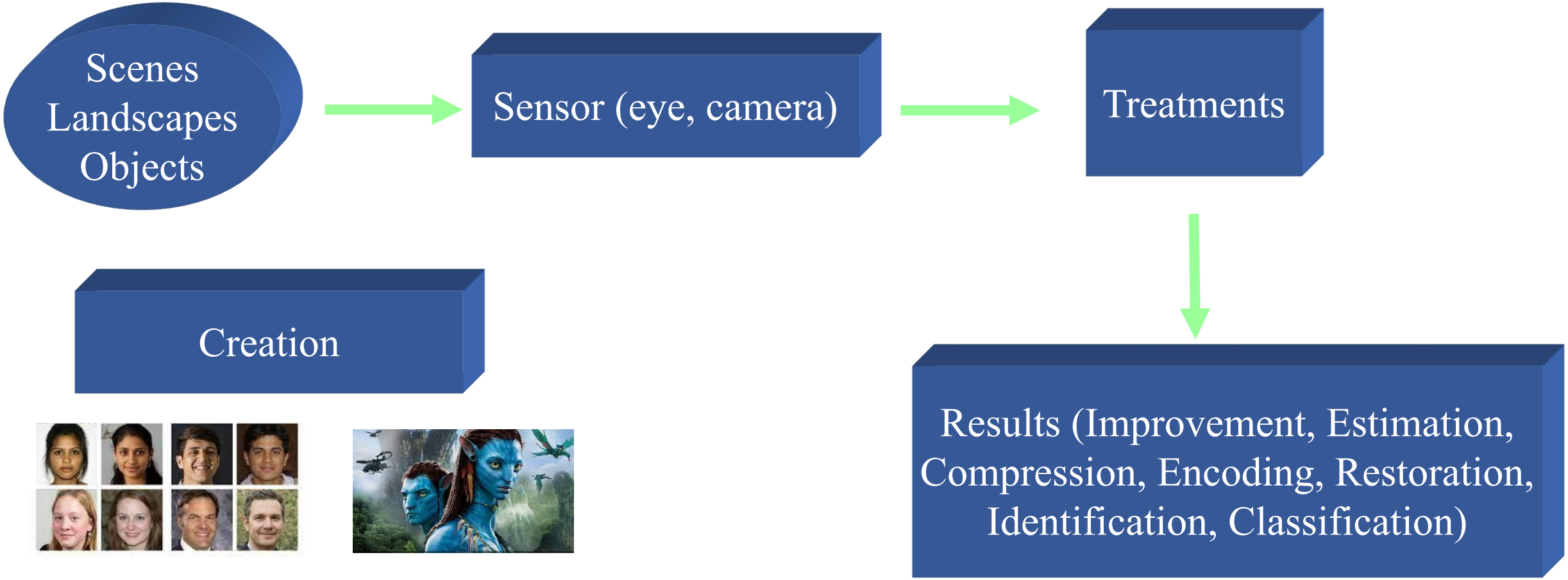
Intro | SPIDER-MAN PLANK | CROSSBODY MOUNTAIN C

Leisure/Commercial: Films, Series, Documentaries , advertising, promotion (vlog)

News (sometimes fake → Manipulation)

Video Surveillance: Street, Roads, Airport, Store, etc.

Introduction

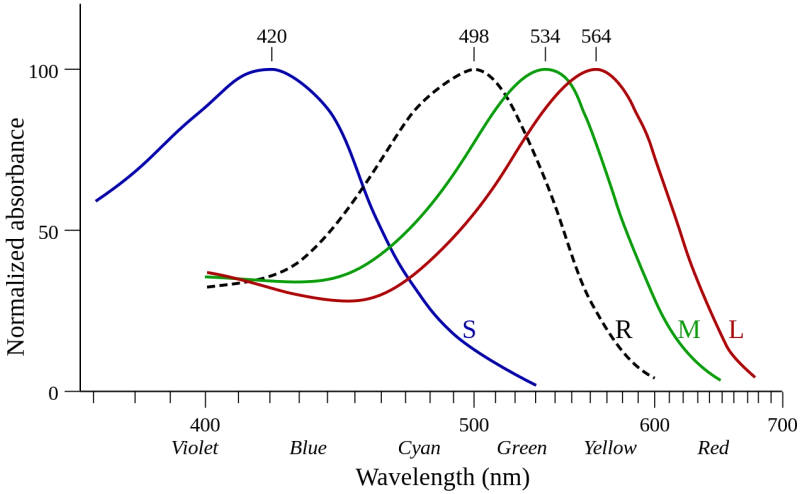


Introduction

Sunlight: electromagnetic waves.

The wavelengths of visible light range from 380 to 780 nm.

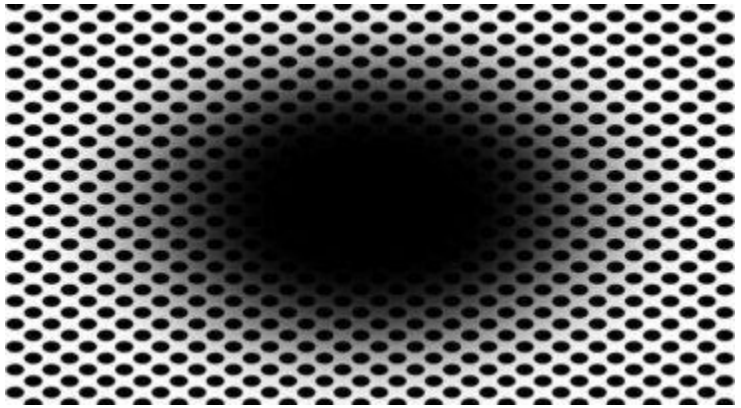
- Retina detects characteristics of light from its photoreceptors:
- the rods (R): insensitive to color and sensitive to movement,
- cones that are sensitive to colors. The cones are of three types conventionally called: blue (S), green (L) and red (M)

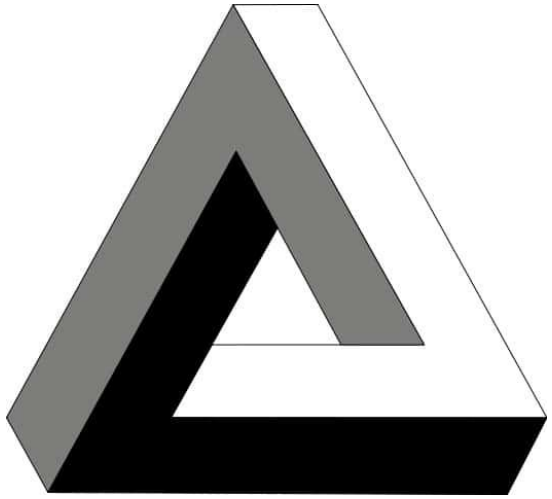


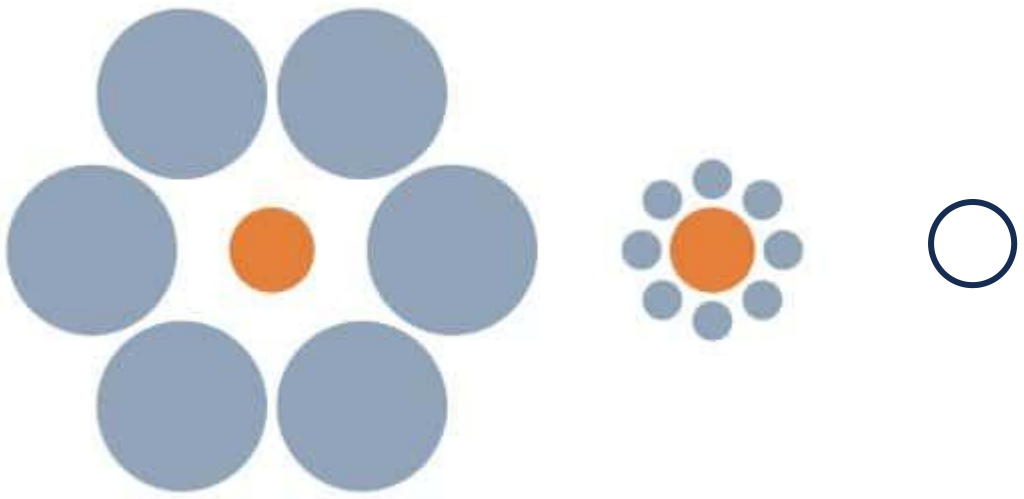
By Vectorized version of the GFDL image Cone-response.png uploaded by User:Maxim Razin based on work by User:DrBob and User:Zeimusu . —After Bowmaker and Dartnall (1980). & quot;Visual pigments of rods and cones in a human retina&quot ;. J. Physiol . 298:501-11. DOI:10.1113/jphysiol.1980.sp013097. PMID 7359434., CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=2447660

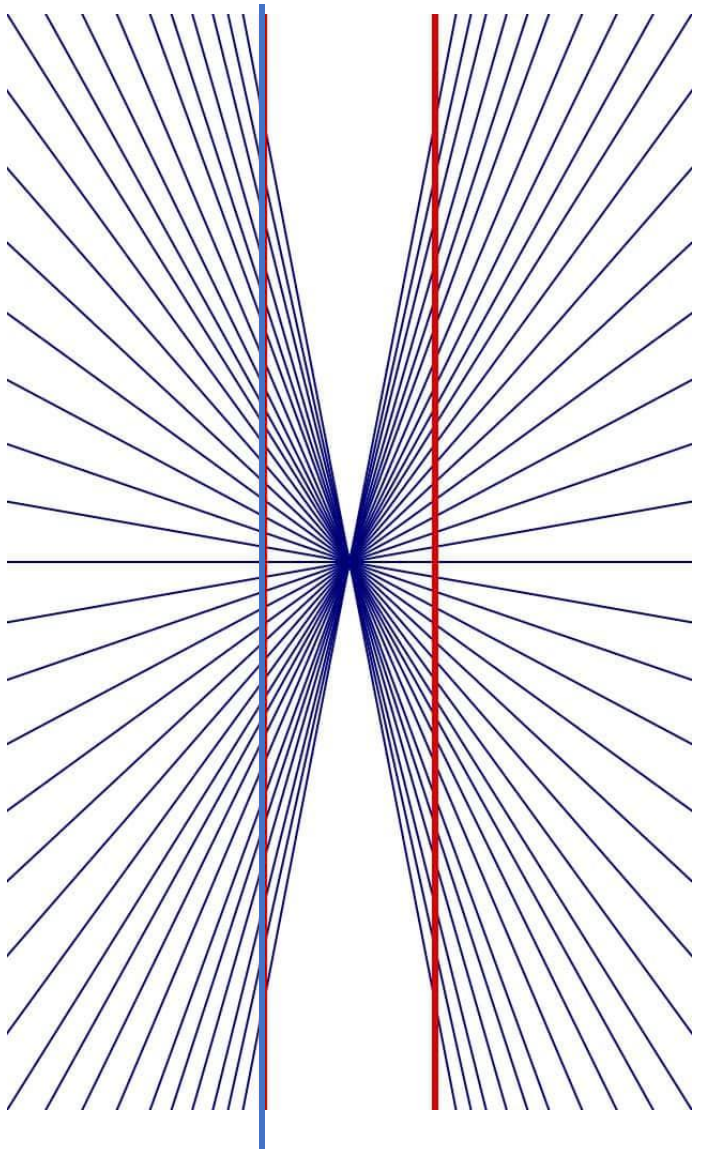
- Brain receives two flat, inverted images, not very colorful, largely blurred.
- Area V1 (primary visual cortex) extracts visual information: color, orientation, movement, contrast, spatial frequency, shape)
- Extraction and Processing by separate, more specialized brain areas: V3 (shape), V4 (color+orientation), V5 (movement)
- Information aggregated and used for recognition of what is seen and/or for action.

Introduction





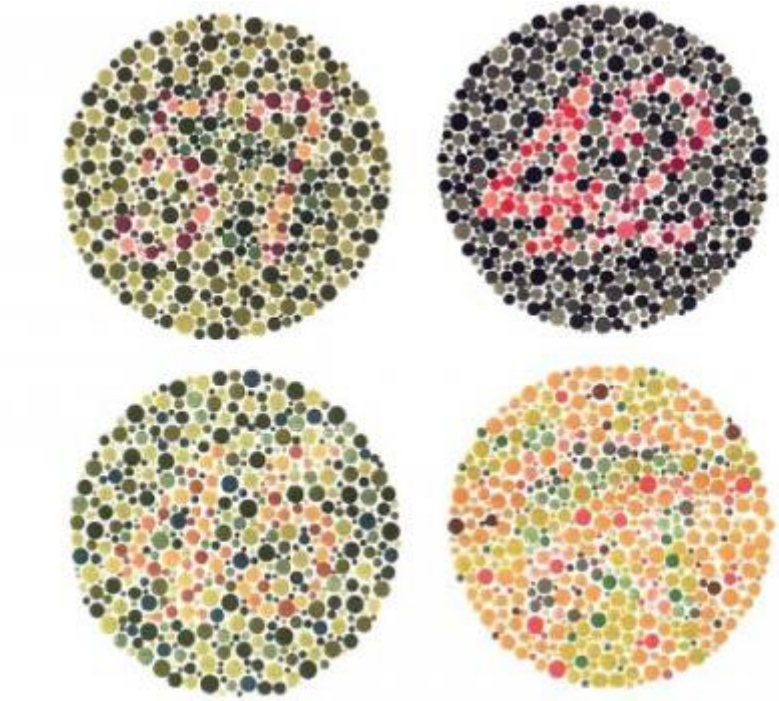




https://www.futura-sciences.com/sciences/photos/photos-top-15-illusions-optique-plus-surprenantes-691/

Introduction

https://illusionoftheyear.com/2009/05/the-illusion-of-sex/

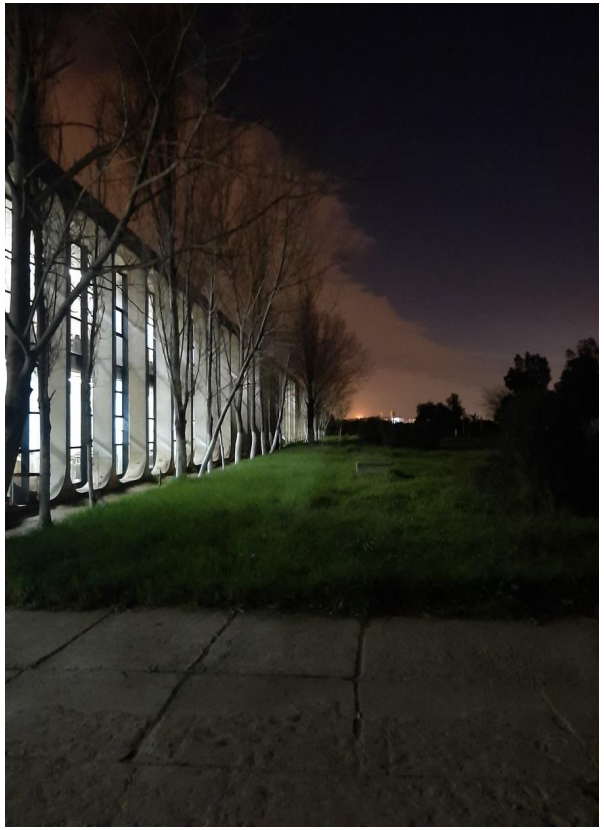


https://www.register-design.com/__trashed-2/

Definitions

Digital image: discrete form of a continuous phenomenon.

- Created , processed, stored in binary form (sequence of *0s* and *1s*).
- Two-dimensional: L rows and C columns
- Information: characteristic of light intensity (color or gray levels).



Definitions

Digital image: discrete form of a continuous phenomenon.

- Created , processed, stored in binary form (sequence of *0s* and *1s*).
- Two-dimensional: L rows and C columns
- Information: characteristic of light intensity (color or gray levels).
- Pixel: “ picture element ", basic unit of the image corresponding to a discretization step.

Position and value (grayscale or color).

176	173	172	174	175	174	175
179	185	187	181	174	173	165
197	181	168	167	171	169	170
161	170	180	183	180	174	175

$$2^3 = 256$$

Definitions

Digital image: discrete form of a continuous phenomenon.

- Created , processed, stored in binary form (sequence of *0s* and *1s*).
- Two-dimensional: L rows and C columns
- Information: characteristic of light intensity (color or gray levels).
- Pixel: “ picture element ", basic unit of the image corresponding to a discretization s

Position and value (grayscale or color).

- Color representation: RGB, HSV, etc.

Blue Channel

51	51	53	54	60	57
15	30	44	53	55	58
89	91	95	99	102	98
53	70	86	95	97	99
101	103	107	110	114	107
65	82	98	107	109	108
73	86	101	113	111	110
113	111	115	116	115	113

60
65
101
104

Green Cha

Red Channel

Definitions

Digital image: discrete form of a continuous phenomenon.

- Created , processed, stored in binary form (sequence of *0s* and *1s*).
- Two-dimensional: L rows and C columns
- Information: characteristic of light intensity (color or gray levels).
- Pixel: “ picture element ", basic unit of the image corresponding to a discretization step.

Position and value (grayscale or color).

- Color representation: RGB, HSV, etc.

$$2^{8^3} \approx 16.\ 10^6$$

0.299 Red + 0.587 Green + 0.114 Blue

couleur	R	G	B	Octet 1	Octet 2	Octet 3
	0	0	0	00000000	00000000	00000000
	255	255	255	11111111	11111111	11111111
	255	0	0	11111111	00000000	00000000
	0	255	0	00000000	11111111	00000000
	0	0	255	00000000	00000000	11111111
	132	122	191	10000100	01111010	10111111

Faculty of Electrical Engineering, USTHB [akourgli@usthb.dz
http://perso.usthb.dz/~akourgli/

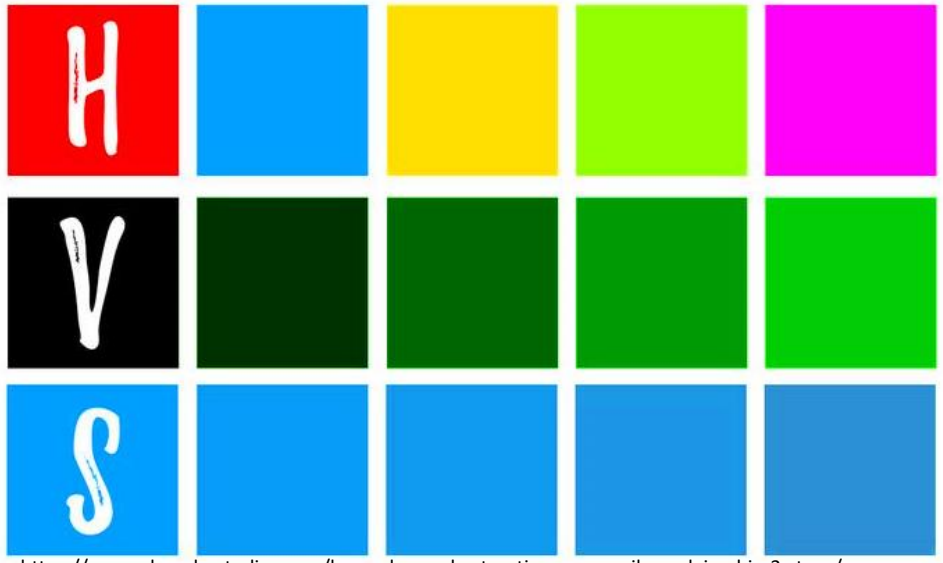
Definitions

Digital image: discrete form of a continuous phenomenon.

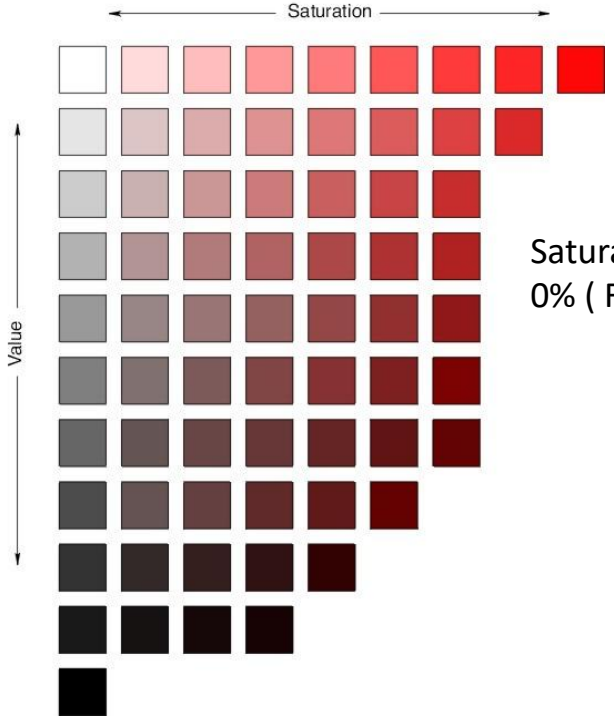
- Created , processed, stored in binary form (sequence of *0s* and *1s*).
- Two-dimensional: L rows and C columns
- Information: characteristic of light intensity (color or gray levels).
- Pixel: “ picture element ", basic unit of the image corresponding to a discretization step.

Position and value (grayscale or color).

- Color representation: RGB, HSV, etc.



https://www.oleanderstudios.com/hue-value-and-saturation-very-easily-explained-in-3-steps/



Saturation (Purity, Intensity)
0% (Fade, Gray) → 100% (Intense)

Value (Brightness)
0% (Dark → 100% (Bright)

https://learn.leighcotnoir.com/artspeak/elements-color/hue-value-saturation/cone_slice/

Definitions

Digital image: discrete form of a continuous phenomenon.

- Created , processed, stored in binary form (sequence of *0s* and *1s*).
- Two-dimensional: L rows and C columns
- Information: characteristic of light intensity (color or gray levels).
- Pixel: “ picture element ", basic unit of the image corresponding to a discretization step.

Position and value (grayscale or color).

- Color representation: RGB, HSV, etc.

Note : Vector Images SEED4NA-logo.pdf SEED4NA-logo.bmp

The curves and mathematical figures of the image are stored

If a vector image is enlarged (or reduced), the distance between paths is recalculated and then adjusted to the correct proportions

Definitions

Digital image: discrete form of a continuous phenomenon.

- Created , processed, stored in binary form (sequence of *0s* and *1s*).
- Two-dimensional: L rows and C columns (***Definition***)
- Information: characteristic of light intensity (color or gray levels).
- Pixel: “ picture element ", basic unit of the image corresponding to a discretization step.

Position and value (grayscale or color).

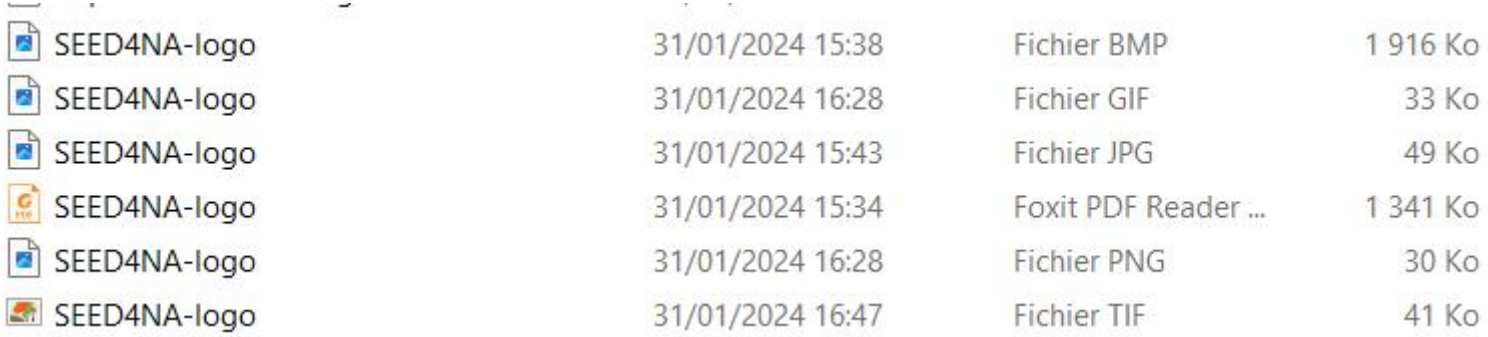
- Color representation: RGB, HSV, etc.

- ***Resolution*** of an image: the number of pixels contained in the image per unit of length. Expressed in **dpi** (dots per inch) or **dpi** (dots per inch). It defines the sharpness and quality of an image.

Definitions

Some Image and Video Formats:

Formats	Benefits	Disadvantages
BMP *(Bitmap):* Graphic image storing pixels in the form of an array of points.	Excellent quality	Large files
GIF (*Graphics Interchange Format*): Small images, icons, buttons on web pages, etc.	Good compression, possibility of animation	Limited to 256 colors
.JPEG .JPG (*Joint Photography Experts Group*): Textured photos and images.	compressed files (>90%)	Loss of quality
PNG (Portable Network Graphics) Intended to replace GIF	Excellent lossless compression	Less effective on large photos
TIFF (Tagged Image File Format)	Lossless compression. Transparency layer.	Large uncompressed files



Definitions

Some Image and Video Formats:





SEED4NA-logo	31/01/2024 15:38	Fichier BMP	1 916 Ko
SEED4NA-logo	31/01/2024 16:28	Fichier GIF	33 Ko
SEED4NA-logo	31/01/2024 15:43	Fichier JPG	49 Ko
SEED4NA-logo	31/01/2024 15:34	Foxit PDF Reader ...	1 341 Ko
SEED4NA-logo	31/01/2024 16:28	Fichier PNG	30 Ko
SEED4NA-logo	31/01/2024 16:47	Fichier TIF	41 Ko

Definitions

Some Image and Video Formats:

Formats	Benefits	Disadvantages
AVI (*Audio Video Interleave*)	Excellent video and audio quality, accepted by many systems	Large file size. inability to take into account multiple audio tracks,
MP4 (*Moving Picture Expert Group*) Standard for the Web	MP4 videos good compromise quality and file sizes	Encoding and decoding processes require a lot of resources.
MKV (*Matroska*)	Additionally integrates video files, audio tracks, menus	Need suitable codec available on the net
MOV Quicktime Apple video montage	Higher quality than MP4 files	Compress video and format it for the web,
Google WebM	Extremely small size for acceptable quality	Not supported by other browsers
WMV (*Windows Media Video*)	Excellent quality while being small	Not compatible with Apple and Linux

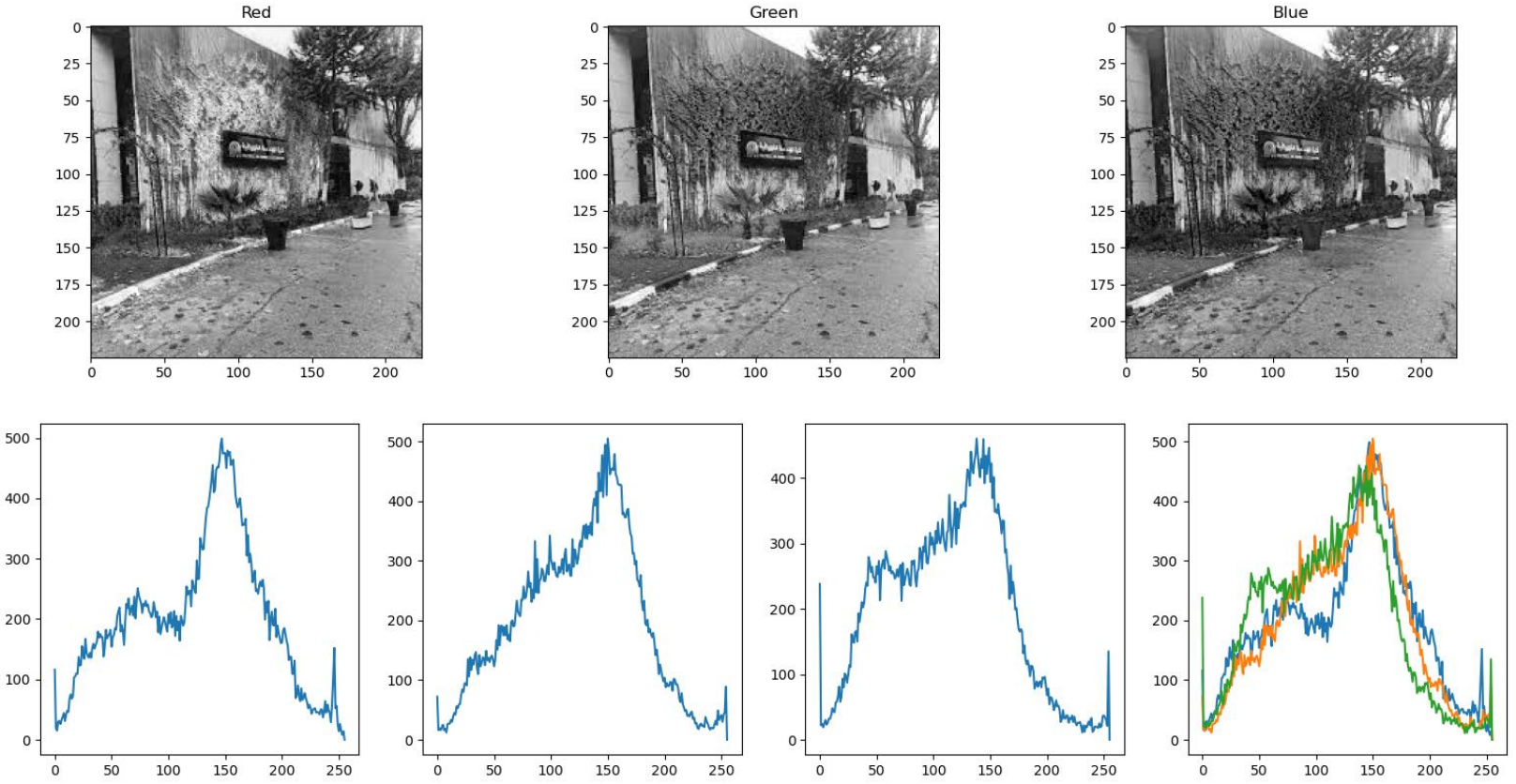
..\ TImages \USTHB.mp4

Basic Operations

- ✓ Histogram concepts
- ✓ Correction of image dynamics using affine transformations on the histogram
- ✓ Spatial filtering and 2D Convolution: mask concepts (average, Gaussian, binomial, etc.)
- ✓ Linear then non-linear smoothing of the image (median, etc.)
- ✓ Edge detection (Objectives, Types of edges, Roberts mask, Prewitt , Sobel , Laplacian operators, Marr-Hildreth filter ,…)
- ✓ Frequency filtering: (2D FFT and separability property, low-pass filter, high-pass filter, etc.)
- ✓ Morphological operations (dilation, erosion, opening, closing, etc.)
- ✓ Thresholding and Binarization

Basic Operations: Histogram

- ✓ Histogram concepts: a discrete function which associates to each intensity value the number of pixels having this value



Basic Operations: Histogram Equalization

✓ Histogram concepts

✓ Correction of image dynamics using affine transformations on the histogram

Histogram equalization is a method of adjusting contrast.

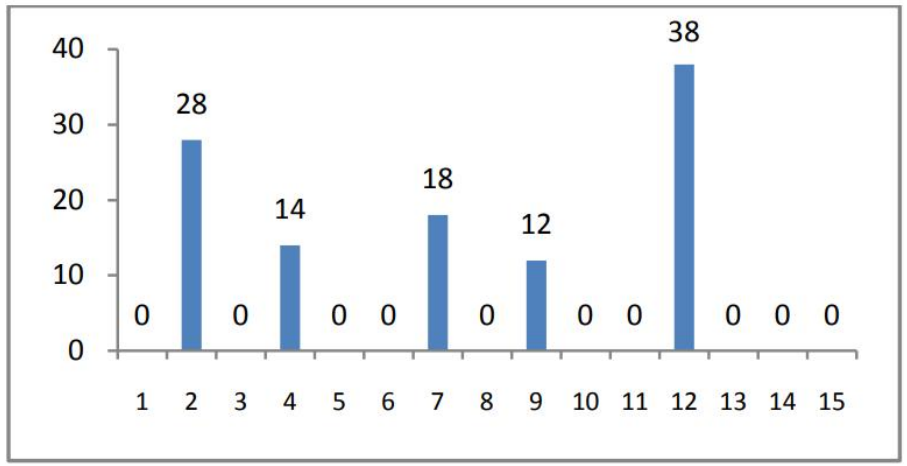
This transformation is constructed from the cumulative histogram of the initial image by ***spreading*** the histogram. A transformation is applied to each pixel of the image, and therefore to obtain a new image from an independent operation on each pixel.

$$T(x_k) = (L-1)\sum_{j=0}^{k} p(x_j) = \frac{(L-1)}{n}\sum_{j=0}^{k} n_j$$

Basic Operations: Histogram Equalization

12	12	12	12	12	12	12	12	12	12
12	9	9	2	2	2	2	9	9	12
12	9	2	7	7	7	7	2	9	12
12	2	7	4	4	4	4	7	2	12
12	2	7	2	4	4	2	7	2	12
12	2	7	4	4	4	4	7	2	12
12	2	7	2	4	4	2	7	2	12
12	2	7	4	2	2	4	7	2	12
12	9	2	7	7	7	7	2	9	12
12	9	9	2	2	2	2	9	9	12
12	12	12	12	12	12	12	12	12	12

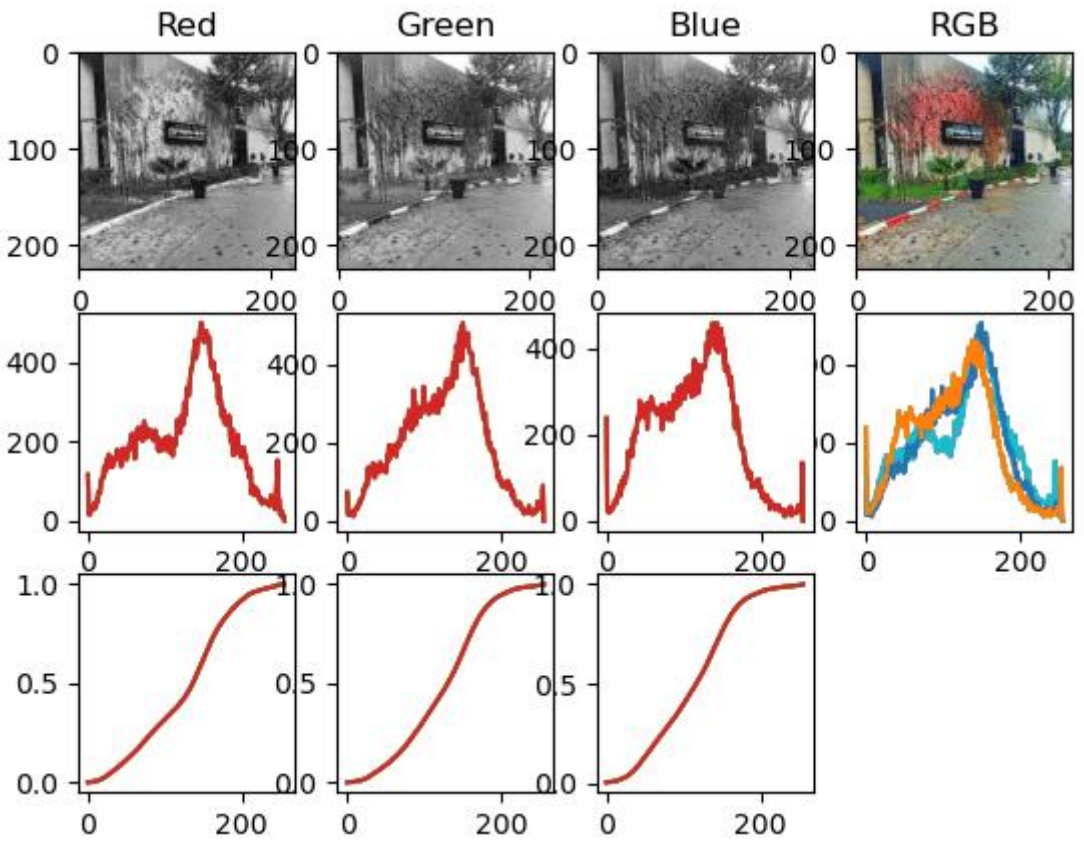
http://virtuelcampus.univ-msila.dz/factech/wp-content/uploads/2020/01/M2-ESEM-Vision-Artif.pdf

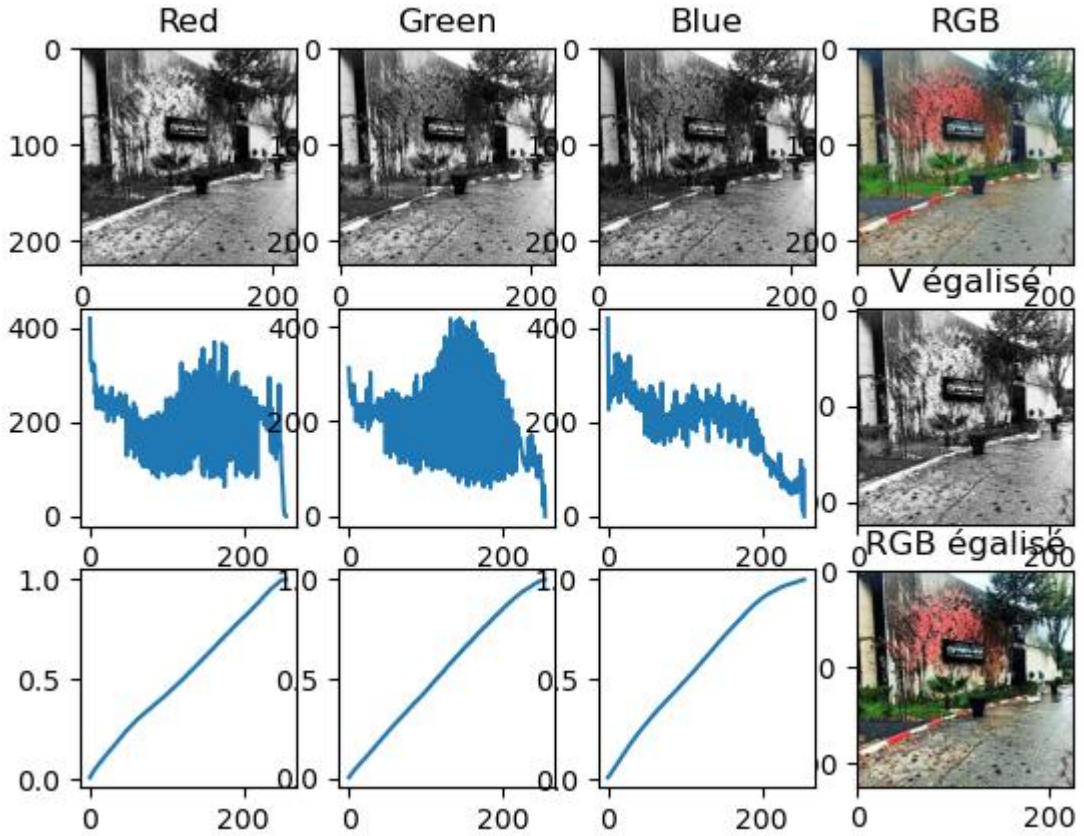


2	4
4	6
7	8
9	10
12	15

Basic Operations: Histogram Equalization

✓ Histogram concepts





Basic Operations: Spatial Filtering

✓ Spatial filtering and 2D Convolution: mask concepts (average, Gaussian, binomial, etc.)

another image having different spatial and frequency properties :

- Linear: Replace the value of each pixel with a weighted average calculated with neighboring pixels. The mask contains the weighting coefficients of each pixel.

$$y(m,n) = \sum_i \sum_j h(i,j)x\,(m-i, n-j)$$

Example

h(-1,-1)	h(-1.0)	h(-1.0)
h(0,-1)	h(0,0)	h(0.1)
h(1,-1)	h(1,0)	h(1,1)

$$y(m,n) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} h(i,j)x(m-i, n-j)$$

x(0,0)	x(0,1)	x(0.2)	x(0.3)	x(0.4)
x(1,0)	x(1,1)	x(1,2)	x(1,3)	x(1,4)
x(2,0)	x(2,1)	x(2,2)	x(2,3)	x(2,4)
x(3,0)	x(3,1)	x(3,2)	x(3,3)	x(3,4)
x(4,0)	x(4,1)	x(4,2)	x(4,3)	x(4,4)

$$y(m,n) = \sum_i \sum_j g(i,j)x\,(m+i, n+j)$$

Applying the filter h(i,j) amounts to scanning the image with the mask g(i,j) = h(-i,-j)

Basic Operations: Spatial Filtering

✓ Spatial filtering and 2D Convolution: mask concepts (average, Gaussian, binomial, etc.)

another image having different spatial and frequency properties :

- Linear: Replace the value of each pixel with a weighted average calculated with neighboring pixels. The mask contains the weighting coefficients of each pixel.

$$y(m,n) = \sum_i \sum_j h(i,j)x\,(m-i, n-j)$$

Example

h(-1,-1)	h(-1.0)	h(-1.0)
h(0,-1)	h(0,0)	h(0.1)
h(1,-1)	h(1,0)	h(1,1)

$$y(m,n) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} h(i,j)x(m-i, n-j)$$

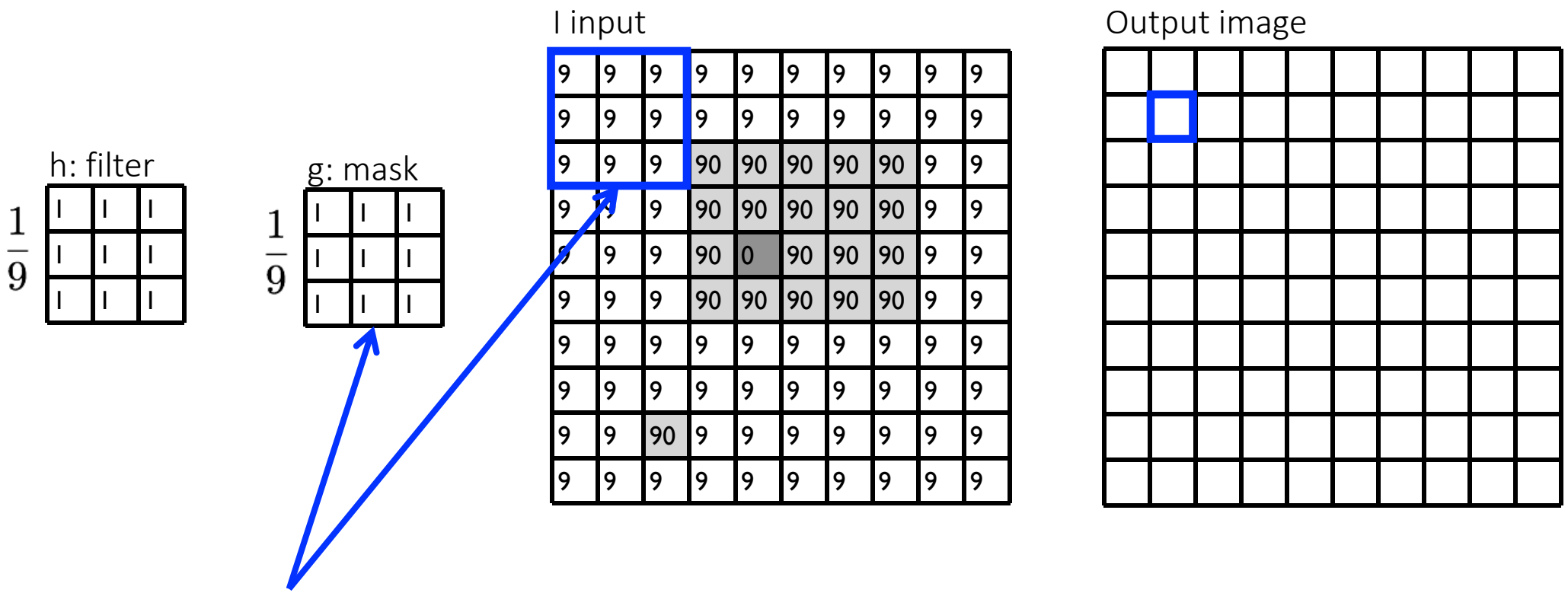
h(1,1)	h(1,0)	h(1,-1)
h(0.1)	h(0,0)	h(0,-1)
h(-1.0)	h(-1.0)	h(-1,-1)

x(0,0)	x(0,1)	x(0.2)	x(0.3)	x(0.4)
x(1,0)	x(1,1)	x(1,2)	x(1,3)	x(1,4)
x(2,0)	x(2,1)	x(2,2)	x(2,3)	x(2,4)
x(3,0)	x(3,1)	x(3,2)	x(3,3)	x(3,4)
x(4,0)	x(4,1)	x(4,2)	x(4,3)	x(4,4)

$$y(m,n) = \sum_i \sum_j g(i,j)x\,(m+i, n+j)$$

Applying the filter h(i,j) amounts to scanning the image with the mask g(i,j) = h(-i,-j)

Basic Operations: Spatial Filtering



Sweep the image with the mask $\quad g(i,j) = h(-i,-j)$

$$y(m,n) = \sum_i \sum_j h(i,j)x\,(m-i, n-j)$$

$$y(m,n) = \sum_i \sum_j g(i,j)x\,(m+i, n+j)$$

Basic Operations: Spatial Filtering

h: filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

g: mask

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9								

Basic Operations: Spatial Filtering

h: filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

g: mask

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9								

Basic Operations: Spatial Filtering

h: filter

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

g: mask

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	18							

Basic Operations: Spatial Filtering

h: filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

g: mask

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	18	27	36	36	36	27	18	
	9								

Basic Operations: Spatial Filtering

h: filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

g: mask

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	18	27	36	36	36	27	18	
	9	27	45	63	63	63	45	27	

Basic Operations: Spatial Filtering

h: filter

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

g: mask

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	18	27	36	36	36	27	18	
	9	27	45	63	63	63	45	27	
	9								

Basic Operations: Spatial Filtering

h: filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

g: mask

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

Medium

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	18	27	36	36	36	27	18	
	9	27	45	63	63	63	45	27	
	9	36	53	80	80	90	63	36	
	9	36	53	80	80	90	63	36	
	9	27	35	53	53	63	45	27	
	9	18	27	36	36	36	27	18	
	18	18	18	9	9	9	9	9	
	18	18	18	9	9	9	9	9	

See annex for a step by step output image calculation

Basic Operations: Spatial Filtering

h: filter

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Medium

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Gaussian

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpening 'Enhancement'

Note: Separable filters

$$\begin{bmatrix} a & b & c \end{bmatrix} \otimes \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} a\alpha & b\alpha & c\alpha \\ a\beta & b\beta & c\beta \\ a\gamma & b\gamma & c\gamma \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} \times \begin{bmatrix} a & b & c \end{bmatrix}.$$

$$\frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \otimes \frac{1}{3}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

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

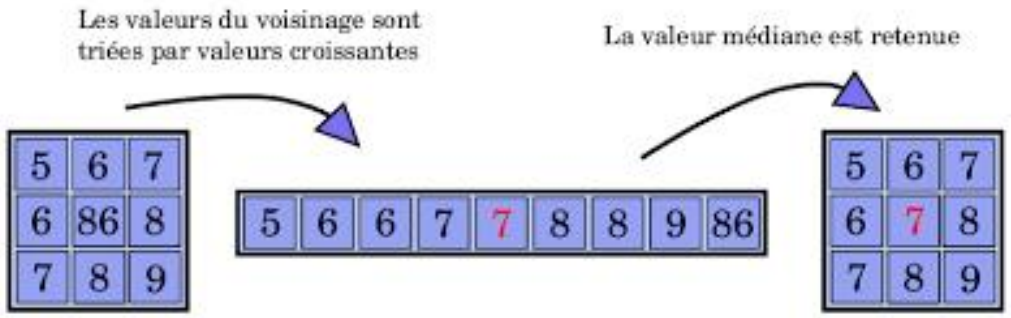
Basic Operations: Spatial Filtering

✓ Spatial filtering and 2D Convolution: mask concepts (average, Gaussian, binomial, etc.)

another image having different spatial and frequency properties :

- Linear: Replace the value of each pixel with a weighted average calculated with neighboring pixels. The mask contains the weighting coefficients of each pixel.

- Nonlinear: *replace the value of each pixel from neighboring pixels* (Median , Bilateral)
-



Basic Operations: Spatial Filtering

✓ Spatial filtering

- Nonlinear: *replace the value of each pixel from neighboring pixels* (Median , Bilateral)
-

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9								

Basic Operations: Spatial Filtering

✓ Spatial filtering

- Nonlinear: *replace the value of each pixel from neighboring pixels* (Median , Bilateral)
-

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	9	9	9	9	9	9	9	

Basic Operations: Spatial Filtering

✓ Spatial filtering

- Nonlinear: *replace the value of each pixel from neighboring pixels* (Median , Bilateral)
-

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	9	9	9	9	9	9	9	
	9	9	9	90	90	90	9	9	

Basic Operations: Spatial Filtering

✓ Spatial filtering

- Nonlinear: *replace the value of each pixel from neighboring pixels* (Median , Bilateral)
-

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	9	9	9	9	9	9	9	
	9	9	9	90	90	90	9	9	
	9	9	90	90	90	90	9	9	
	9	9	90	90	90	90	9	9	
	9	9	9	90	90	90	9	9	
	9	9	9	9	9	9	9	9	
	9	9	9	9	9	9	9	9	
	9	9	9	9	9	9	9	9	

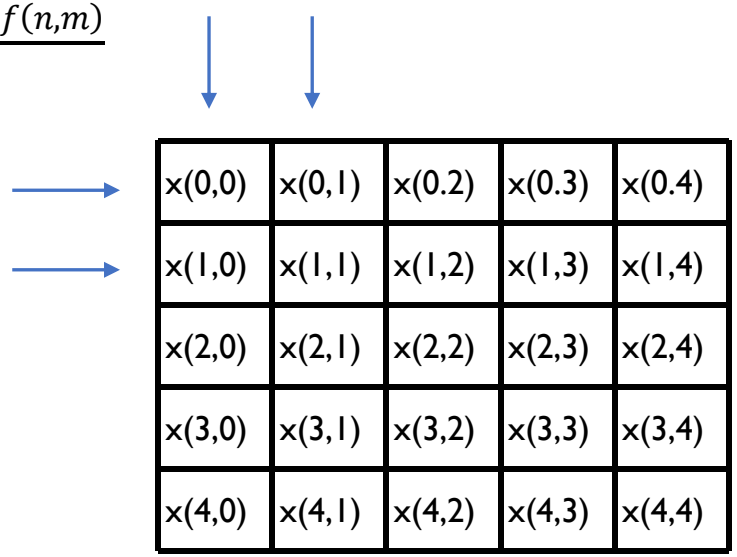
Basic Operations: Spatial Filtering

✓ Spatial filtering: Gradients

Continuous first derivative: $f'(t) = \lim_{u \to 0} \frac{f(t+u) - f(t)}{u}$ → Discrete: Finite differences $f'(n) = \frac{f(n+k) - f(n)}{k} = \frac{f(n) - f(n-k)}{k}$

Image: 2 Dimensions $d_x f(n,m) = \frac{f(n+k,m) - f(n,m)}{k}$ or $d_y f(n,m) = \frac{f(n,m+k) - f(n,m)}{k}$

k = 1 → 1/2 pixel offset → $h = [-1 \;\; 1]$



Basic Operations: Spatial Filtering

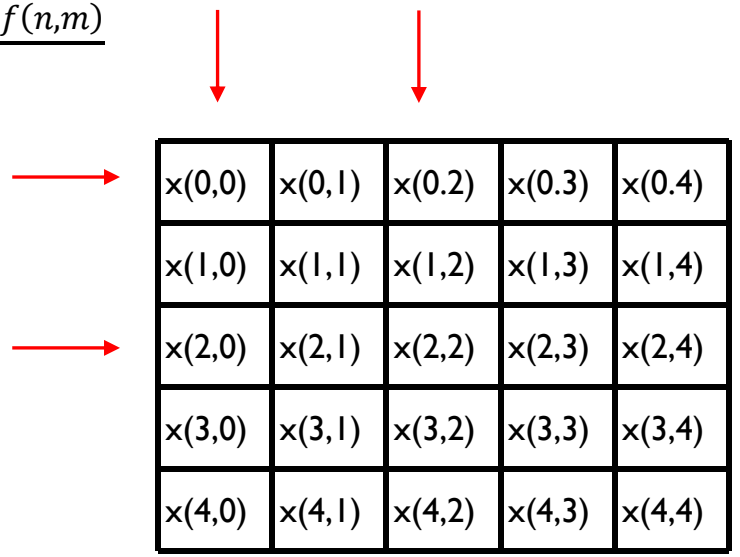
✓ Spatial filtering: Gradients

Continuous first derivative: $f'(t) = \lim_{u \to 0} \frac{f(t+u) - f(t)}{u}$ → Discrete: Finite differences $f'(n) = \frac{f(n+k) - f(n)}{k} = \frac{f(n) - f(n-k)}{k}$

Image: 2 Dimensions $d_x f(n,m) = \frac{f(n+k,m) - f(n,m)}{k}$ or $d_y f(n,m) = \frac{f(n,m+k) - f(n,m)}{k}$

k = 1 → 1/2 pixel offset → $h = [-1 \;\; 1]$

k = 2 → $h = [-1 \;\; 0 \;\; 1]$ → $g = [1 \;\; 0 \;\; -1]$



Basic Operations: Spatial Filtering

✓ Spatial filtering: Gradients

Continuous first derivative: $f'(t) = \lim_{u \to 0} \frac{f(t+u) - f(t)}{u}$ → Discrete: Finite differences $f'(n) = \frac{f(n+k) - f(n)}{k} = \frac{f(n) - f(n-k)}{k}$

Image: 2 Dimensions $d_x f(n,m) = \frac{f(n+k,m) - f(n,m)}{k}$ or $d_y f(n,m) = \frac{f(n,m+k) - f(n,m)}{k}$

k = 1 → 1/2 pixel offset → $h = [-1 \;\; 1]$

k = 2 → $h = [-1 \;\; 0 \;\; 1]$ → $g = [1 \;\; 0 \;\; -1]$

g: Filter

1	0	-1
1	0	- 1
1	0	-1

Horizontal

=

1
1
1

Smoothing

·

1	0	- 1

Derivation

1	1	1
0	0	0
- 1	-1	-1

Vertical

x(0,0)	x(0,1)	x(0.2)	x(0.3)	x(0.4)
x(1,0)	x(1,1)	x(1,2)	x(1,3)	x(1,4)
x(2,0)	x(2,1)	x(2,2)	x(2,3)	x(2,4)
x(3,0)	x(3,1)	x(3,2)	x(3,3)	x(3,4)
x(4,0)	x(4,1)	x(4,2)	x(4,3)	x(4,4)

Basic Operations: Spatial Filtering

✓ Spatial Filtering and 2D Convolution: Gradients

h: filter

$\frac{1}{3}$

- 1	0	1
- 1	0	1
- 1	0	1

g: mask

$\frac{1}{3}$

1	0	- 1
1	0	- 1
1	0	- 1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	0								

Sweep the image with the mask

$g(i,j) = h(-i,-j)$

$$y(m,n) = \sum_i \sum_j g(i,j)x(m+i,n+j)$$

Basic Operations: Spatial Filtering

✓ Spatial Filtering and 2D Convolution: Gradients

h: filter

$\frac{1}{3}$

-1	0	1
-1	0	1
-1	0	1

g: mask

$\frac{1}{3}$

1	0	-1
1	0	-1
1	0	-1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	0	-27							

Sweep the image with the mask

$g(i,j) = h(-i,-j)$

$$y(m,n) = \sum_i \sum_j g(i,j)x(m+i,n+j)$$

Basic Operations: Spatial Filtering

✓ Spatial Filtering and 2D Convolution: Gradients

h: filter

$\frac{1}{3}$

- 1	0	1
- 1	0	1
- 1	0	1

g: mask

$\frac{1}{3}$

1	0	- 1
1	0	- 1
1	0	- 1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	0	-27	-27						

Sweep the image with the mask

$g(i,j) = h(-i,-j)$

$$y(m,n) = \sum_i \sum_j g(i,j)x(m+i,n+j)$$

Basic Operations: Spatial Filtering

✓ Spatial Filtering and 2D Convolution: Gradients

h: filter

$\frac{1}{3}$

-1	0	1
-1	0	1
-1	0	1

g: mask

$\frac{1}{3}$

1	0	-1
1	0	-1
1	0	-1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	0	-27	-27	0					

Sweep the image with the mask

$g(i,j) = h(-i,-j)$

$$y(m,n) = \sum_i \sum_j g(i,j)x(m+i,n+j)$$

Basic Operations: Spatial Filtering

✓ Spatial Filtering and 2D Convolution: Gradients

h: filter

$\frac{1}{3}$

-1	0	1
-1	0	1
-1	0	1

g: mask

$\frac{1}{3}$

1	0	-1
1	0	-1
1	0	-1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	0	-27	-27	0	0	0	27	27	

Sweep the image with the mask

$$g(i,j) = h(-i,-j)$$

$$y(m,n) = \sum_i \sum_j g(i,j)x(m+i,n+j)$$

Basic Operations: Spatial Filtering

✓ Spatial Filtering and 2D Convolution: Gradients

h: filter

$\frac{1}{3}$

-1	-1	-1
-0	0	0
1	1	1

g: mask

$\frac{1}{3}$

1	1	1
0	0	0
-1	-1	-1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	0								

Sweep the image with the mask

$g(i,j) = h(-i,-j)$

$$y(m,n) = \sum_i \sum_j g(i,j)x(m+i,n+j)$$

Basic Operations: Spatial Filtering

✓ Spatial Filtering and 2D Convolution: Gradients

h: filter

$\frac{1}{3}$

- 1	0	1
- 1	0	1
- 1	0	1

g: mask

$\frac{1}{3}$

1	0	- 1
1	0	- 1
1	0	- 1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	0	-27							

Sweep the image with the mask

$g(i,j) = h(-i,-j)$

$$y(m,n) = \sum_i \sum_j g(i,j)x(m+i,n+j)$$

Basic Operations: Spatial Filtering

✓ Spatial Filtering and 2D Convolution: Gradients

h: filter

$\frac{1}{3}$

-1	0	1
-1	0	1
-1	0	1

g: mask

$\frac{1}{3}$

1	0	-1
1	0	-1
1	0	-1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	0	-27	-54						

Sweep the image with the mask

$g(i,j) = h(-i,-j)$

$$y(m,n) = \sum_i \sum_j g(i,j)x(m+i,n+j)$$

Basic Operations: Spatial Filtering

✓ Spatial Filtering and 2D Convolution: Gradients

h: filter

$\frac{1}{3}$

-1	0	1
-1	0	1
-1	0	1

g: mask

$\frac{1}{3}$

1	0	-1
1	0	-1
1	0	-1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	0	-27	-27	-81					

Sweep the image with the mask

$$g(i,j) = h(-i,-j)$$

$$y(m,n) = \sum_i \sum_j g(i,j)x(m+i,n+j)$$

Basic Operations: Spatial Filtering

✓ Spatial Filtering and 2D Convolution: Gradients

h: filter

$\frac{1}{3}$

-1	0	1
-1	0	1
-1	0	1

g: mask

$\frac{1}{3}$

1	0	-1
1	0	-1
1	0	-1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	0	-27	-27	-54	-81	-81	-54	-27	

Sweep the image with the mask

$g(i,j) = h(-i,-j)$

$$y(m,n) = \sum_i \sum_j g(i,j)x(m+i,n+j)$$

Basic Operations: Spatial Filtering

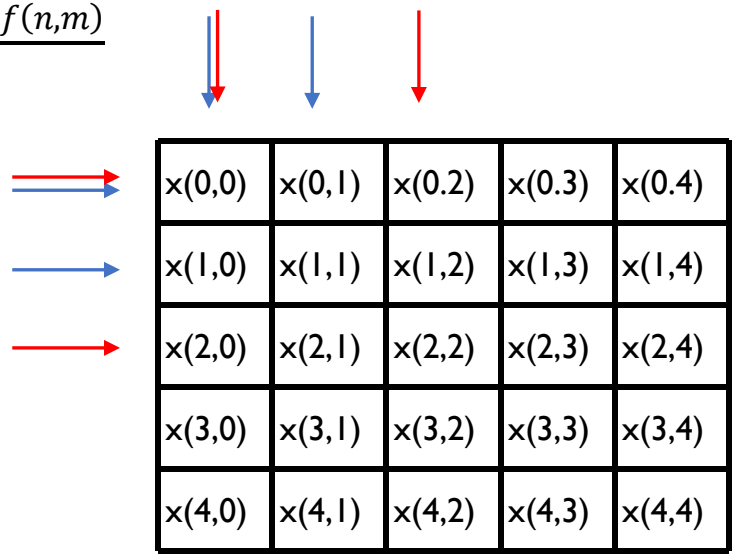
✓ Spatial filtering: Gradients

Continuous first derivative: $f'(t) = \lim_{u \to 0} \frac{f(t+u) - f(t)}{u}$ → Discrete: Finite differences $f'(n) = \frac{f(n+k) - f(n)}{k}$

Image: 2 Dimensions $d_x f(n,m) = \frac{f(n+k,m) - f(n,m)}{k}$ or $d_y f(n,m) = \frac{f(n,m+k) - f(n,m)}{k}$

k = 1 → 1/2 pixel offset → $h = [-1 \;\; 1]$

k = 2 → $h = [-1 \;\; 0 \;\; 1]$ → $g = [1 \;\; 0 \;\; -1]$



g: Filter

1	0	-1
1	0	- 1
1	0	-1

Horizontal

=

1
1
1

Smoothing

·

1	0	- 1

Derivation

1	1	1
0	0	0
- 1	-1	-1

Vertical

Basic Operations: Spatial Filtering

✓ Spatial filtering: Gradients

Continuous first derivative: $f'(t) = \lim_{u \to 0} \frac{f(t+u) - f(t)}{u}$ → Discrete: Finite differences $f'(n) = \frac{f(n+k) - f(n)}{k}$

Image: 2 Dimensions $d_x f(n,m) = \frac{f(n+k,m) - f(n,m)}{k}$ or $d_y f(n,m) = \frac{f(n,m+k) - f(n,m)}{k}$

k = 1 → 1/2 pixel offset → $h = [-1 \;\; 1]$

k = 2 → $h = [-1 \;\; 0 \;\; 1]$ → $g = [1 \;\; 0 \;\; -1]$

x(0,0)	x(0,1)	x(0.2)	x(0.3)	x(0.4)
x(1,0)	x(1,1)	x(1,2)	x(1,3)	x(1,4)
x(2,0)	x(2,1)	x(2,2)	x(2,3)	x(2,4)
x(3,0)	x(3,1)	x(3,2)	x(3,3)	x(3,4)
x(4,0)	x(4,1)	x(4,2)	x(4,3)	x(4,4)

g: Filter

1	0	-1
2	0	-2
1	0	-1

Horizontal

=

1
2
1

Smoothing

·

1	0	- 1

Derivation

1	2	1
0	0	0
- 1	-2	-1

Vertical

Basic Operations: Spatial Filtering

✓ Spatial filtering: Gradients

Continuous first derivative: $f'(t) = \lim_{u \to 0} \frac{f(t+u) - f(t)}{u}$ → Discrete: Finite differences $f'(n) = \frac{f(n+k) - f(n)}{k} = \frac{f(n) - f(n-k)}{k}$

Image: 2 Dimensions $d_x f(n,m) = \frac{f(n+k,m) - f(n,m)}{k} = \frac{f(n,m) - f(n-k,m)}{k}$ Or $d_y f(n,m) = \frac{f(n,m+k) - f(n,m)}{k} = \frac{f(n,m) - f(n,m-k)}{k}$

Second derivative :

Horizontal → $\Delta_x f(n,m) = d_x(d_x f(n,m)) = d_x\left(\frac{f(n,m) - f(n-k,m)}{k}\right) = \frac{\left(f(n+k,m) - f(n,m)\right) - \left(f(n,m) - f(n-k,m)\right)}{k^2} = \frac{f(n+k,m) - 2f(n,m) + f(n-k,m)}{k^2}$

Vertical → $\Delta_y f(n,m) = d_y(d_y f(n,m)) = d_y\left(\frac{f(n,m) - f(n,m-k)}{k}\right) = \frac{\left(f(n,m+k) - f(n,m)\right) - \left(f(n,m) - f(n,m-k)\right)}{k^2} = \frac{f(n,m+k) - 2f(n,m) + f(n,m-k)}{k^2}$

Basic Operations: Spatial Filtering

✓ Spatial filtering: Gradients

Second derivative:

Horizontal → $\Delta_x f(n,m) = d_x(d_x f(n,m)) = d_x\left(\frac{f(n,m)-f(n-k,m)}{k}\right) = \frac{\left(f(n+k,m)-f(n,m)\right)-\left(f(n,m)-f(n-k,m)\right)}{k^2} = \frac{f(n+k,m)-2f(n,m)+f(n-k,m)}{k^2}$

k=1 → $\Delta_x f(n,m) = f(n+1,m) - 2f(n,m) + f(n-1,m) \rightarrow h = [1 \quad -2 \quad 1]$

k=1 → $\Delta_x f(n,m) = f(n,m+1) - 2f(n,m) + f(n,m-1) \rightarrow h = [1 \quad -2 \quad 1]^t$

$\frac{1}{4}$

0	1	0
1	-4	1
0	1	0

$\frac{1}{4}$

Laplacian Filter

Basic Operations: Spatial Filtering

✓ Spatial filtering: Gradients

Second derivative:

Horizontal → $\Delta_x f(n,m) = d_x(d_x f(n,m)) = d_x\left(\frac{f(n,m)-f(n-k,m)}{k}\right) = \frac{\left(f(n+k,m)-f(n,m)\right)-\left(f(n,m)-f(n-k,m)\right)}{k^2} = \frac{f(n+k,m)-2f(n,m)+f(n-k,m)}{k^2}$

k=1 → $\Delta_x f(n,m) = f(n+1,m) - 2f(n,m) + f(n-1,m) \rightarrow h = [1 \quad -2 \quad 1]$

k=1 → $\Delta_x f(n,m) = f(n,m+1) - 2f(n,m) + f(n,m-1) \rightarrow h = [1 \quad -2 \quad 1]^t$

$\frac{1}{4}$

0	-1	0
-1	4	-1
0	-1	0

$\frac{1}{8}$

-1	-1	-1
-1	8	-1
-1	-1	-1

Laplacian Filter

$\frac{1}{4}$

1	-2	1
-2	4	-2
1	-2	1

Gaussian Laplacian

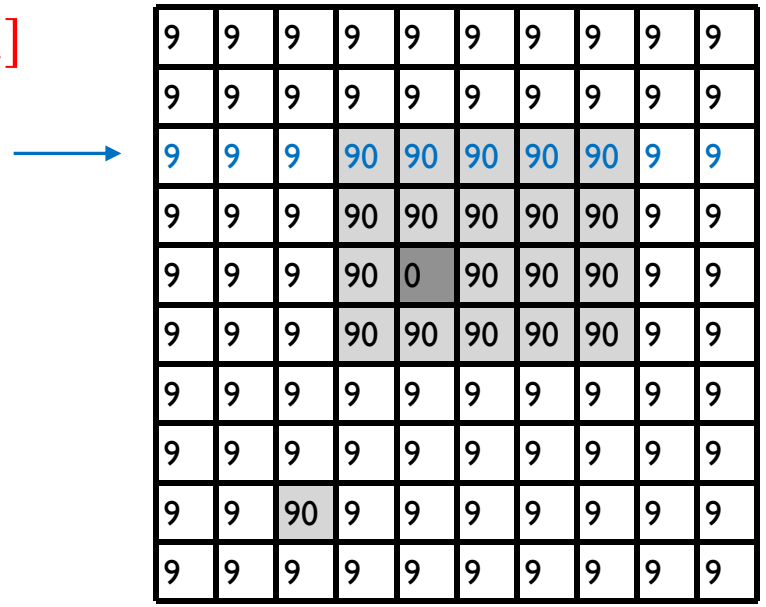
Basic Operations: Spatial Filtering

✓ Spatial filtering: Gradient and Laplacian

First derivative and second derivative:

k = 2 → $h = [-1 \;\; 0 \;\; 1]$ → $g = [1 \;\; 0 \;\; -1]$

k=1 → $h = g = [1 \;\; -2 \;\; 1]$



Basic Operations: Spatial Filtering

✓ Spatial filtering: Gradient and Laplacian

First derivative and second derivative:

k = 2 → $h = [-1 \;\; 0 \;\; 1]$ → $g = [1 \;\; 0 \;\; -1]$

k=1 → $h = g = [1 \;\; -2 \;\; 1]$

	0	-81	-81	0	0	0	81	81	

→

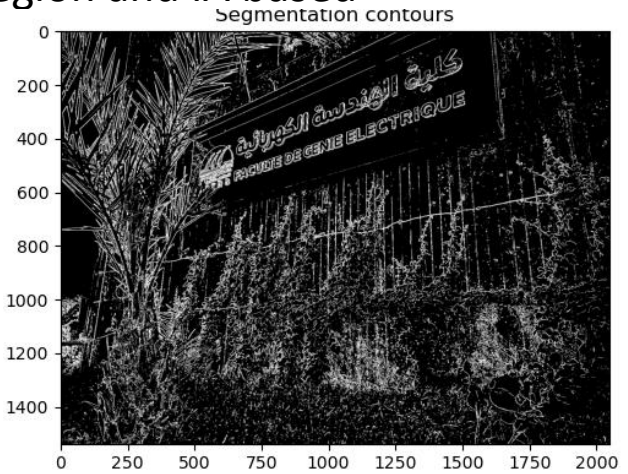
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

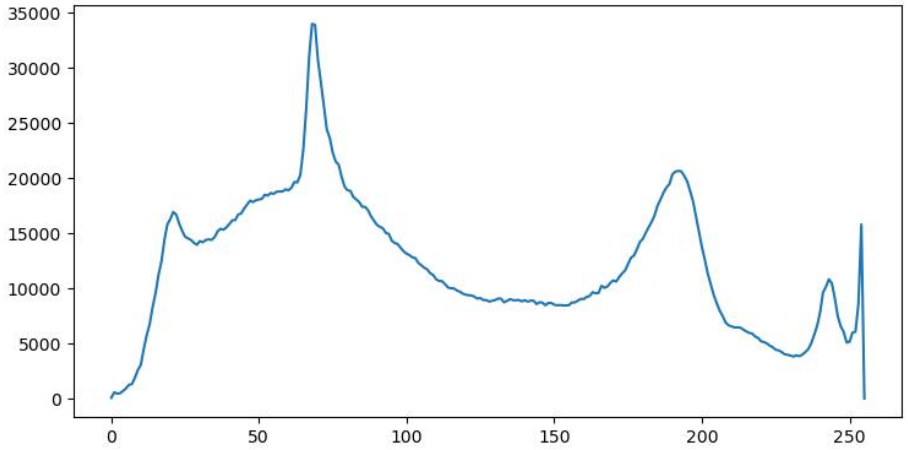
	0	81	-81	0	0	0	-81	63	

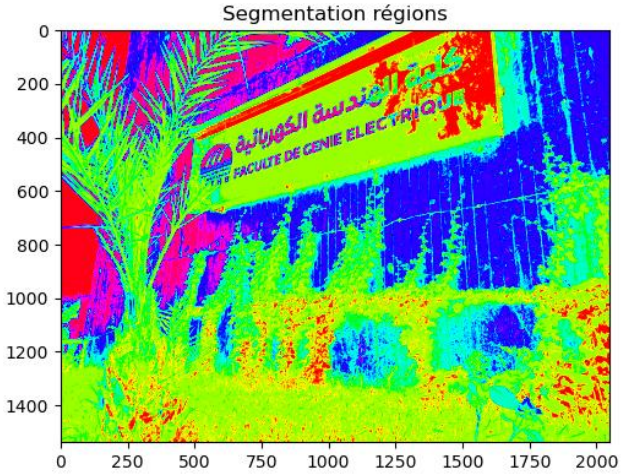
Segmentation

Image segmentation is the process of partitioning a digital image into multiple image regions or image objects (sets of pixels). It is typically used to locate objects and boundaries (lines, curves, etc.) in images. A label is assigned to pixels sharing common characteristics.

Approaches : classical : thresholding, edge, clustering/classification, region and IA based



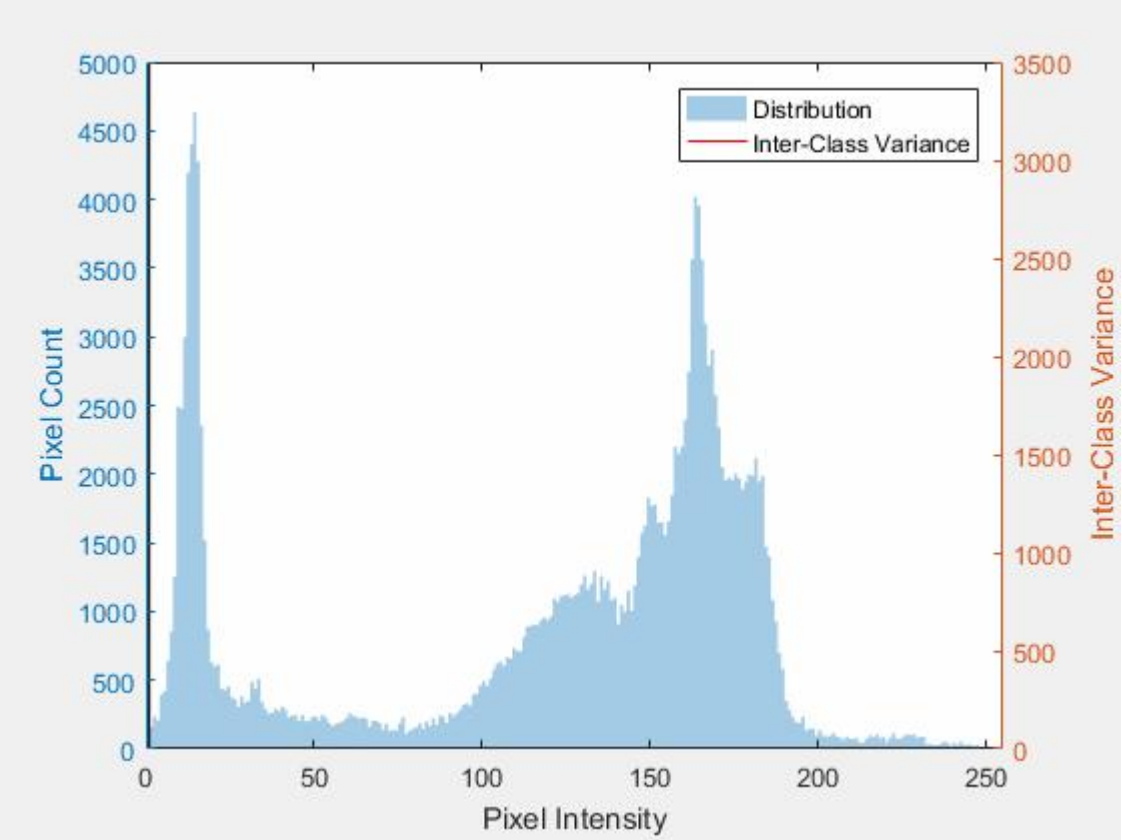




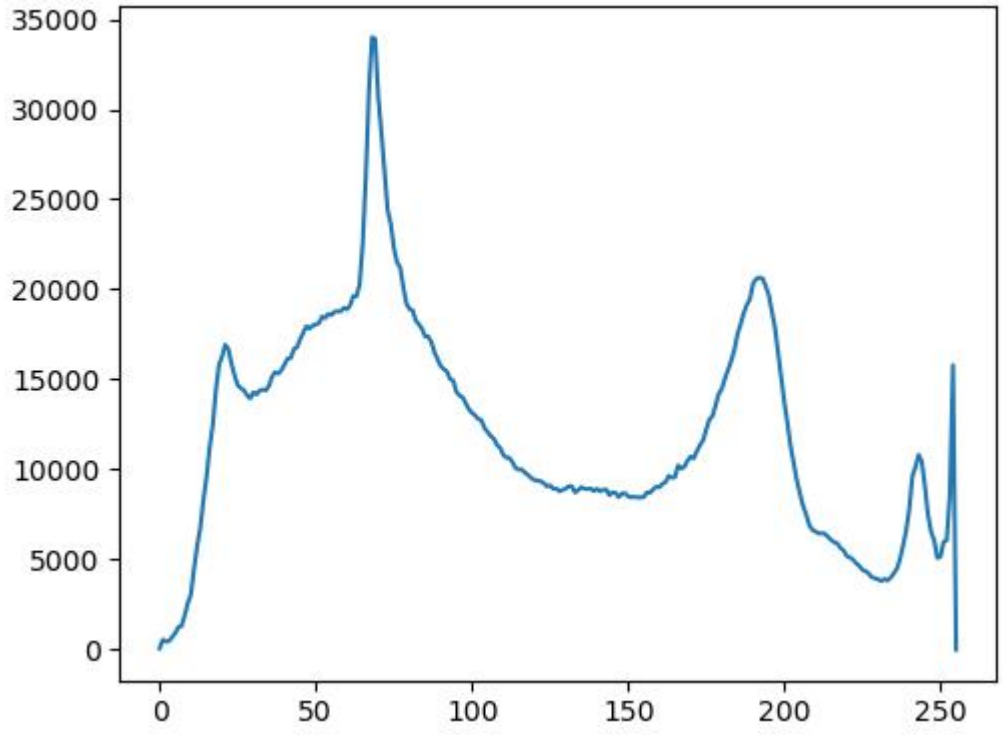
Segmentation using color

Multi-Otsu thresholding is an extension of Otsu's thresholding method, which is used to segment an image into multiple classes or regions based on pixel intensity values.

The image is divided into K classes. The algorithm iteratively searches for thresholds that maximize the variance between the classes, leading to the best separation of intensity levels.



https://en.wikipedia.org/wiki/Otsu%27s_method#/media/File:Otsu's_Method_Visualization.gif



85
160

55
101
155
211

Segmentation

Kmeans is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group**.**

Input :

Specify K number of clusters

Training Set (Data matrix)

Begin

Choose K points randomly (one row of the data matrix). These points are the centers of th clusters (called centroïd).

REPETE

Compute the sum of the squared distance between data points and all centroids.

Assign each point (element of the data matrix) to the closet cluster (centroid)

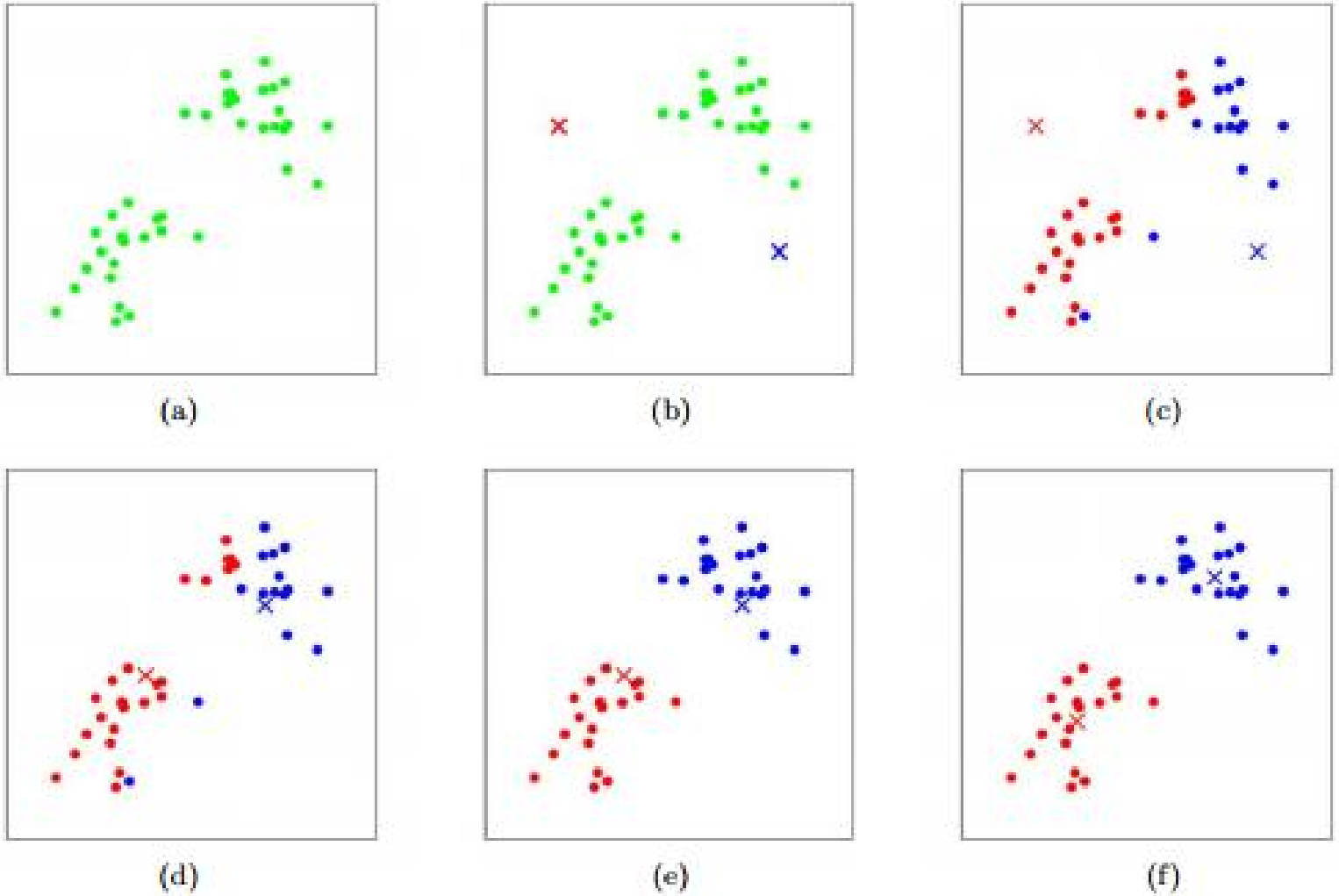
Re Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

UNTIL there is no change to the centroids. i.e assignment of data points to clusters isn't changing.

End

Segmentation

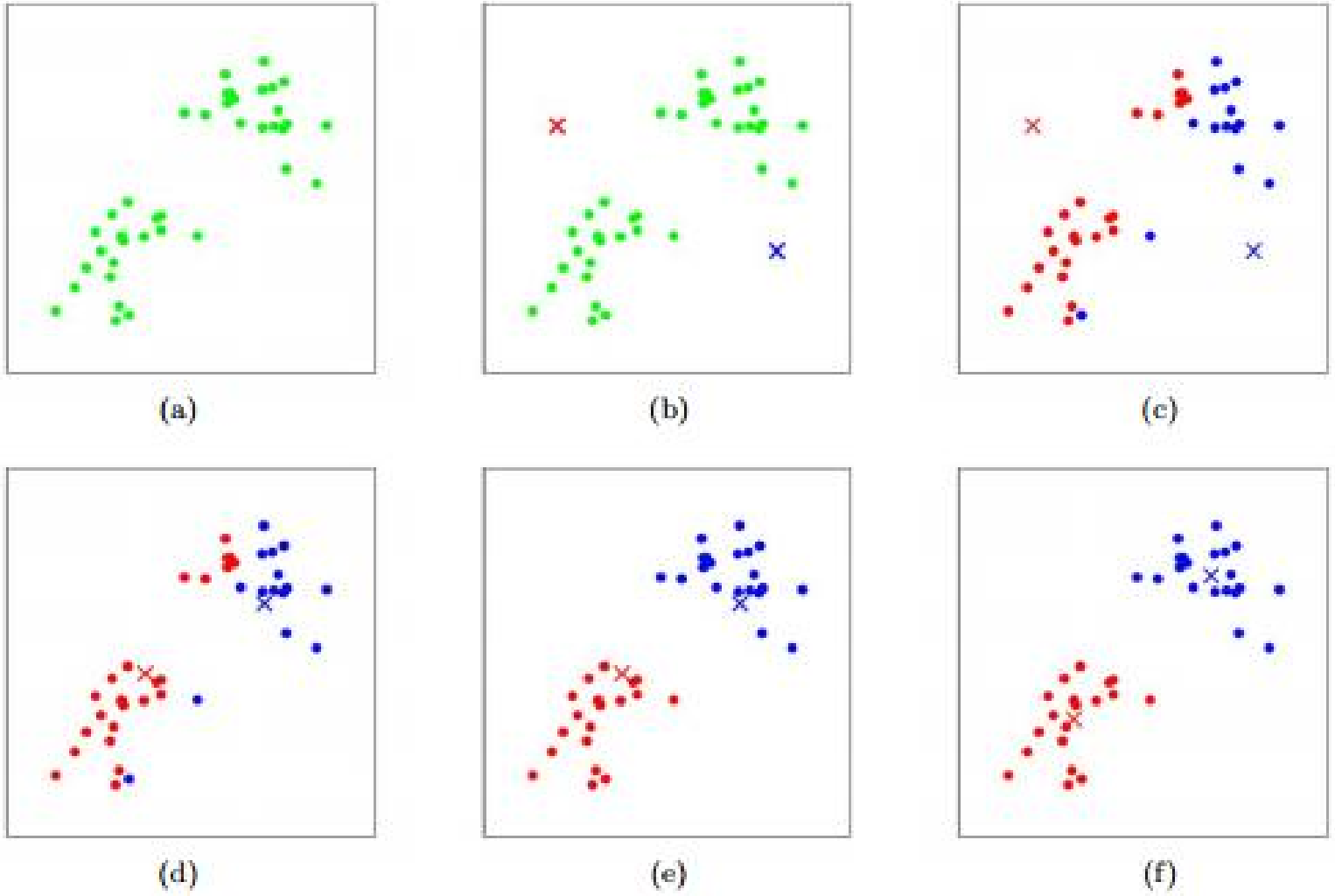
Kmeans is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group**.**



https://stanford.edu/~cpiech/cs221/handouts/kmeans.html

Segmentation

Kmeans is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group**.**



Applications :

Segmentation,

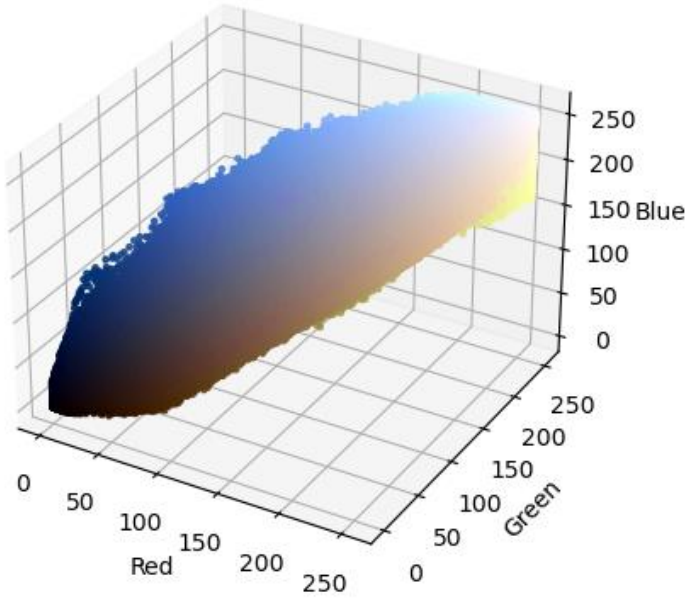
Quantification

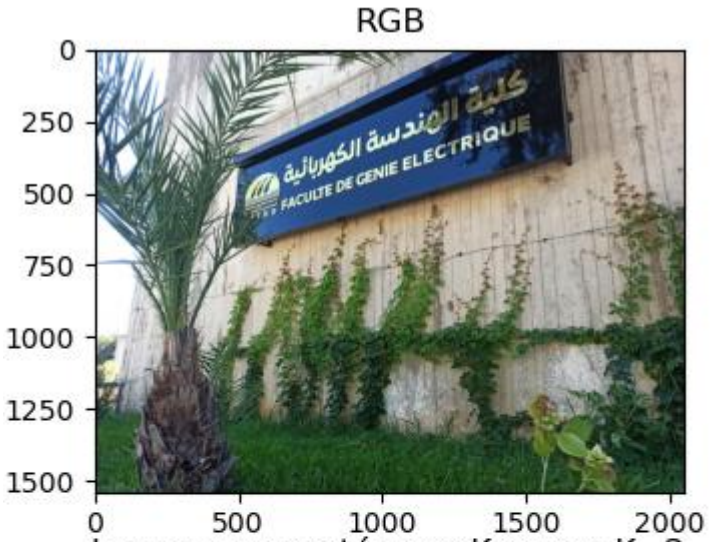
Compression

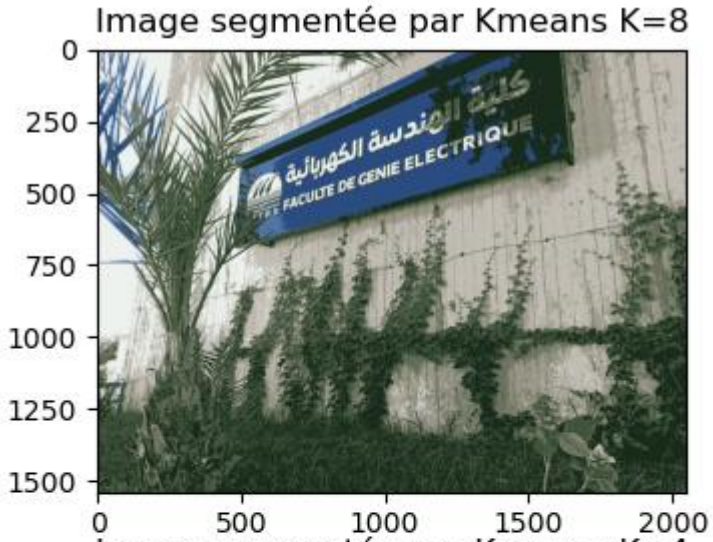
https://stanford.edu/~cpiech/cs221/handouts/kmeans.html

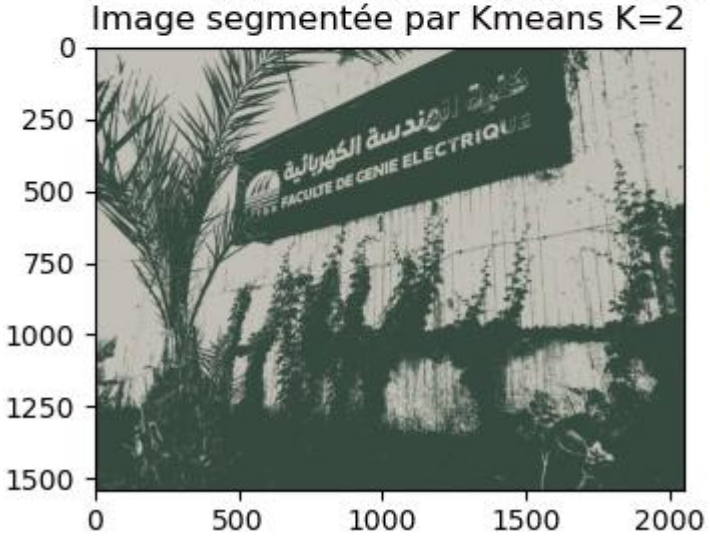
Segmentation

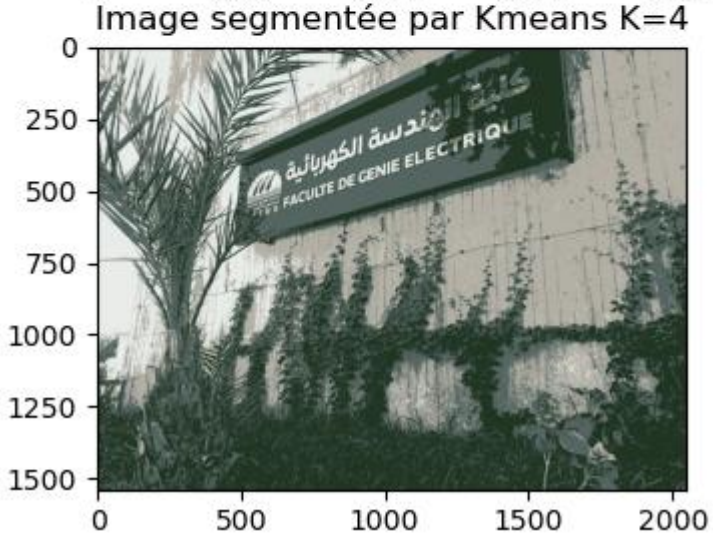
Kmeans is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group**.**











Applications :

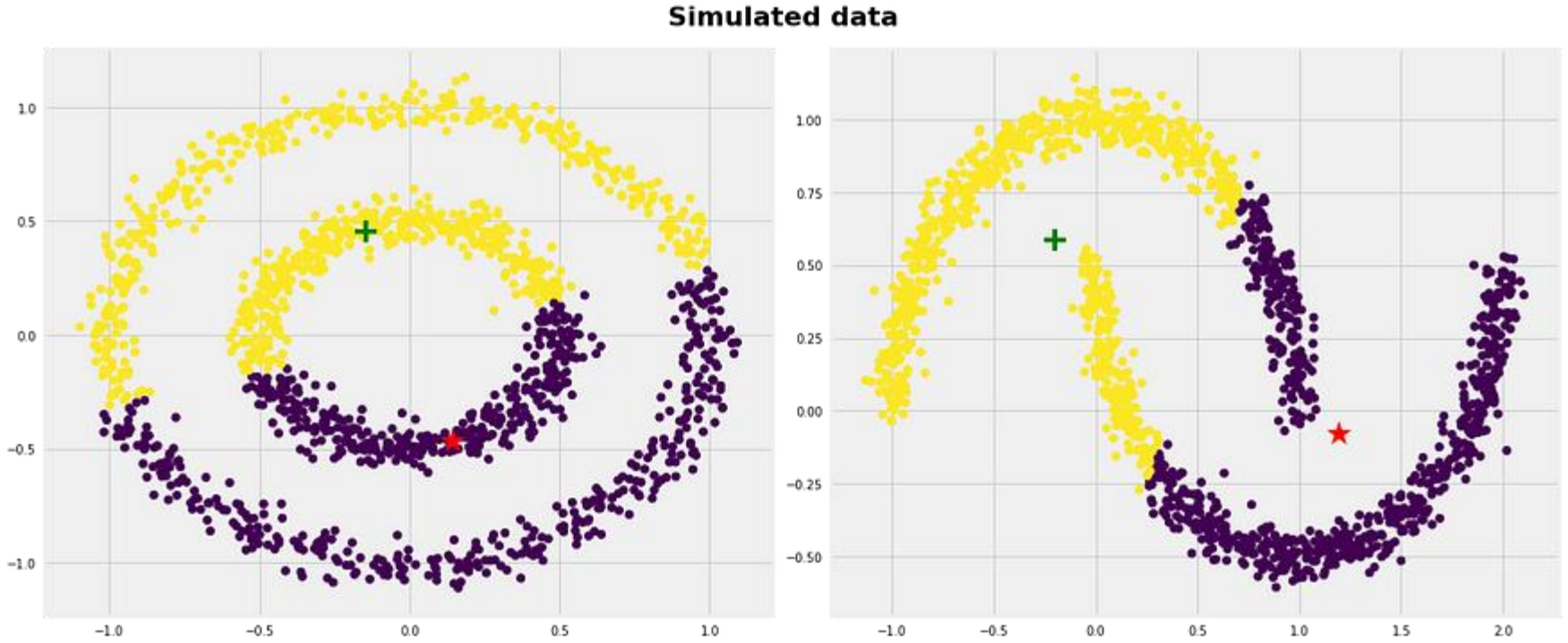
Segmentation, Quantification, Compression

Améliorations : Utilisation d'un noyau, autre distance

Faculty of Electrical Engineering, USTHB [akourgli@usthb.dz
http://perso.usthb.dz/~akourgli/

Segmentation

Kmeans is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group**.**



https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a

Use cases

Customer segmentation based on certain criteria such as purchasing habits or demographics.
In data mining, clustering is used when mining data to identify similar individuals.

Segmentation

Mean-Shift is a non-parametric algorithm for partitioning multidimensional data. It is an iterative algorithm which aims to make a point converge towards the nearest local maximum.

It can be used for segmentation, by immersing the image in a 5-dimensional space, where each pixel is represented by a point having as coordinates its position in x, in y, and its R, G, B values.

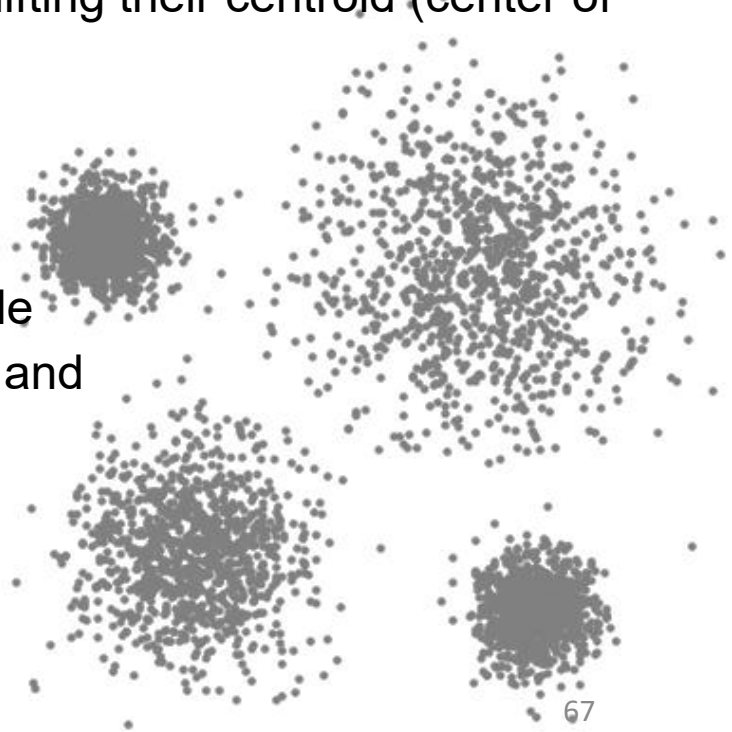
Algorithm :

1. Create a sliding window/cluster for each data-point in feature space
2. Each of the sliding windows is shifted towards higher density regions by shifting their centroid (center of the sliding window) to the data-points' mean within the sliding window.
3. This step will be repeated until no shift yields a higher density

(number of points in the sliding window)

4. Selection of sliding windows by deleting overlapping windows. When multiple sliding windows overlap, the window containing the most points is preserved, and the others are deleted.
5. Assigning the data points to the sliding window in which they reside.

https://www.chioka.in/meanshift-algorithm-for-the-rest-of-us-python/



Segmentation

Mean-Shift is a non-parametric algorithm for p
which aims to make a point converge towards th

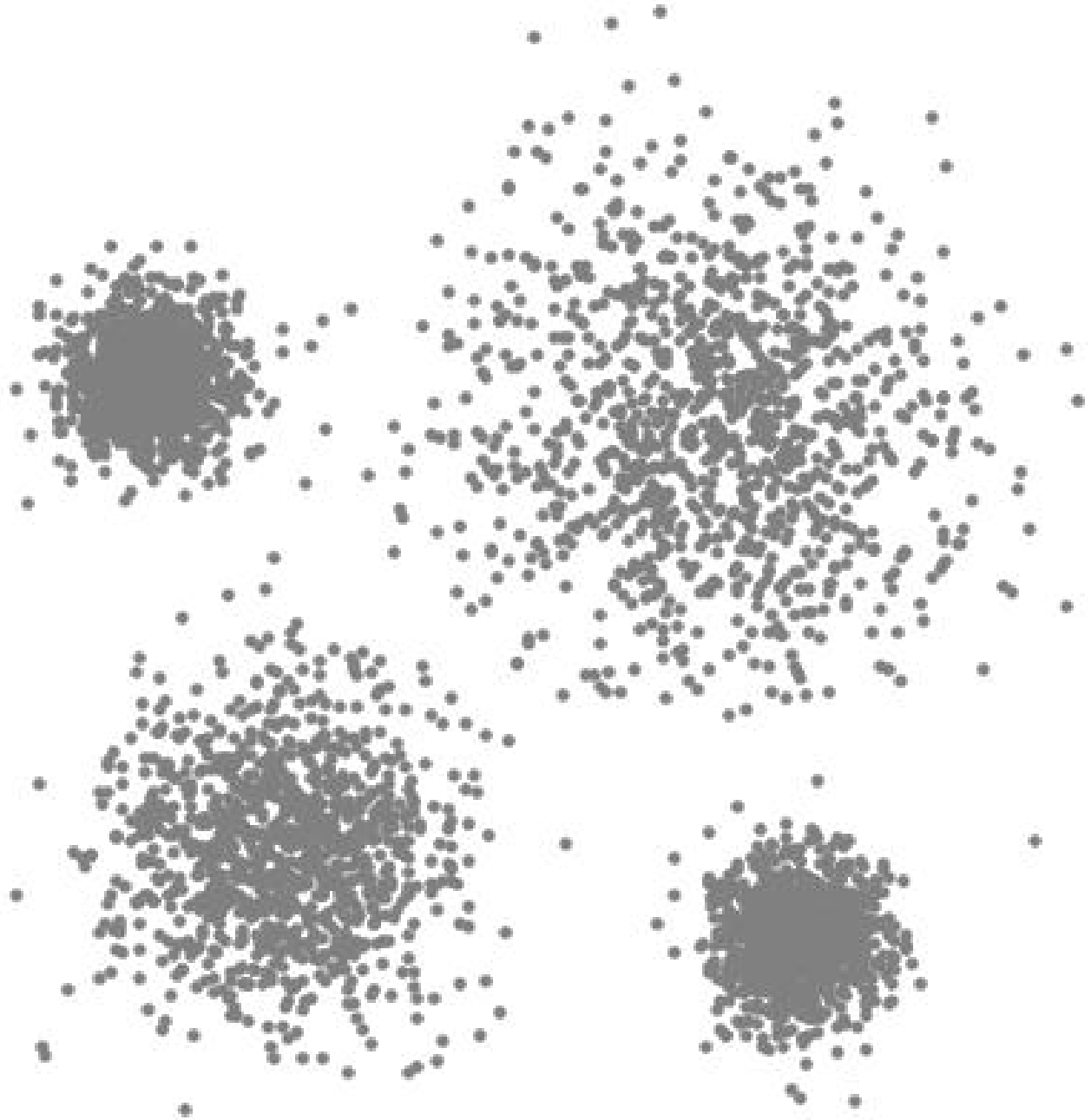
It can be used for segmentation, by immersin
represented by a point having as coordinates its

Algorithm :

1. Create a sliding window/cluster for each data
2. Each of the sliding windows is shifted toward
   the sliding window) to the data-points' mean
3. This step will be repeated until no shift yields
   (number of points in the sliding window)

4. Selection of sliding windows by deleting overl
   sliding windows overlap, the window containing
   the others are deleted.
5. Assigning the data points to the sliding windo

https://www.chioka.in/meanshift-algorithm-for-the



https://sbme-tutorials.github.io/2018/cv/notes/8_week8.html

Faculty of Electrical Engineering, USTHB [akourgli@usthb.dz
http://perso.usthb.dz/~akourgli/

Segmentation

Mean Shift shifts the windows to a higher density region by shifting their centroid (center of the sliding window) to the mean of the data-points inside the sliding window. We can also look at this by thinking of our data points as a probability density function.

Higher density regions correspond to regions with more data-points, and lower density regions correspond to regions with fewer points. Mean Shift tries to find the high-density regions by continually shifting the sliding window closer to the peak of the region. This is also known as hill climbing.

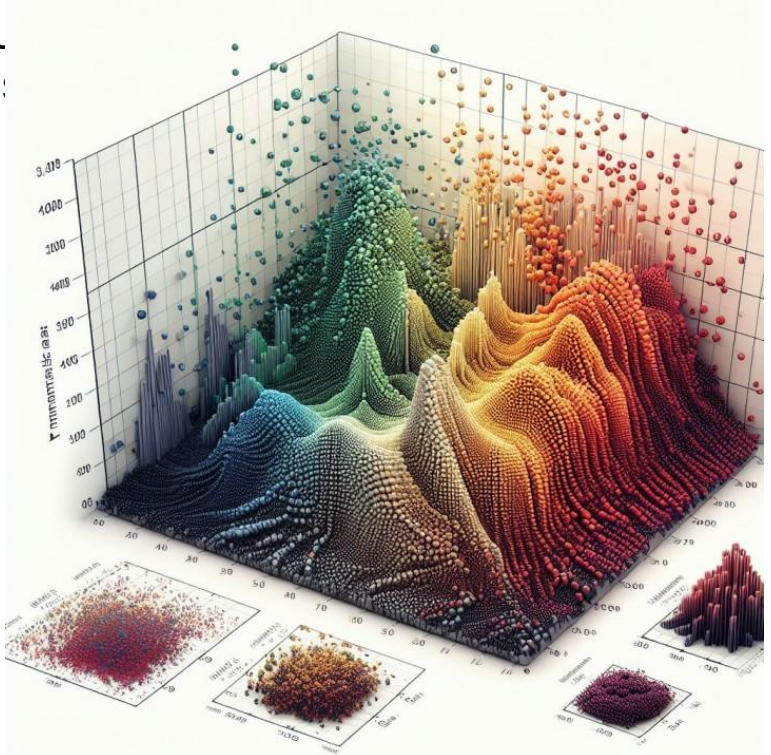
The densest area of the data is determined by the kernel function, which is a fu
the data points based on their distance from the mean. The kernel function us
usually a Gaussian function.

Choosing the right bandwidth / radius

Depending on the bandwidth, the resulting clusters can look quite different.

Extremely small bandwidth → Each point having its own cluster.

Huge bandwidth → one cluster containing all the data-points.



Faculty of Electrical Engineering, USTHB [akourgli@usthb.dz
http://perso.usthb.dz/~akourgli/

Segmentation

Mean Shift shifts the windows to a hi
window) to the mean of the data-points
data points as a probability density func

Higher density regions correspond to re
regions with fewer points. Mean Shift t
window closer to the peak of the region

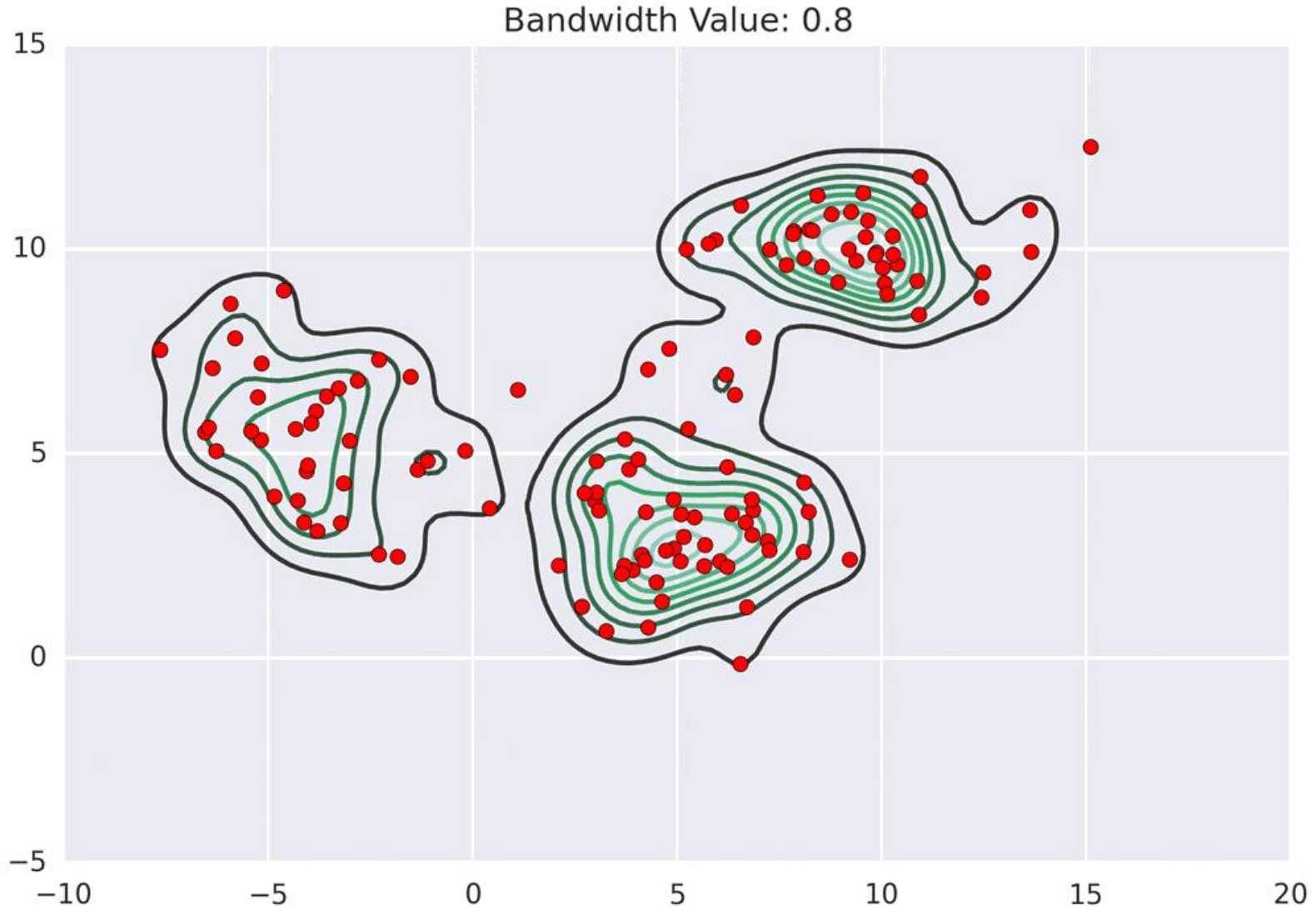
The densest area of the data is determi
the data points based on their distance
usually a Gaussian function.

Choosing the right bandwidth / radiu

Depending on the bandwidth, the result

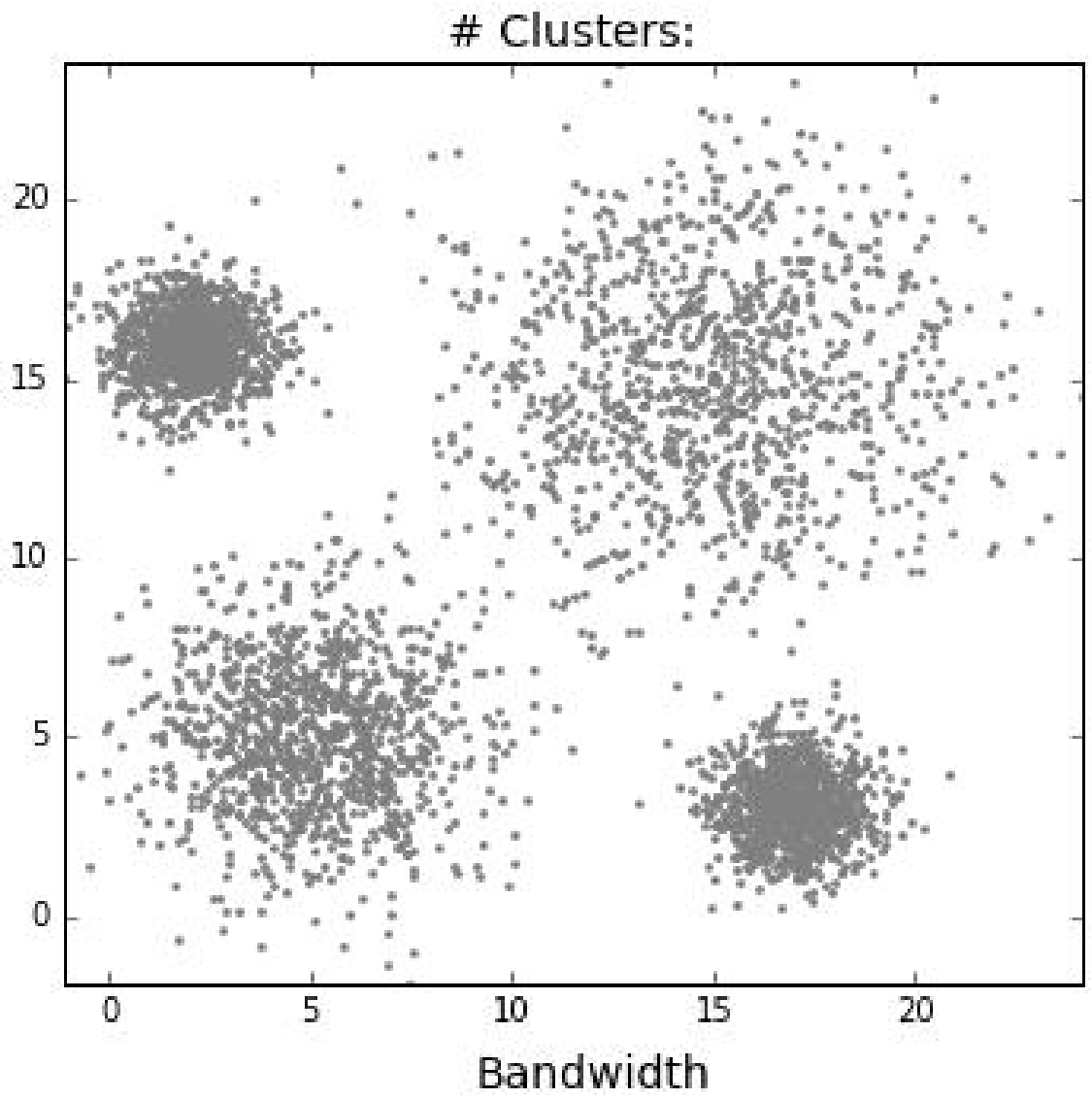
Extremely small bandwidth → Each poi

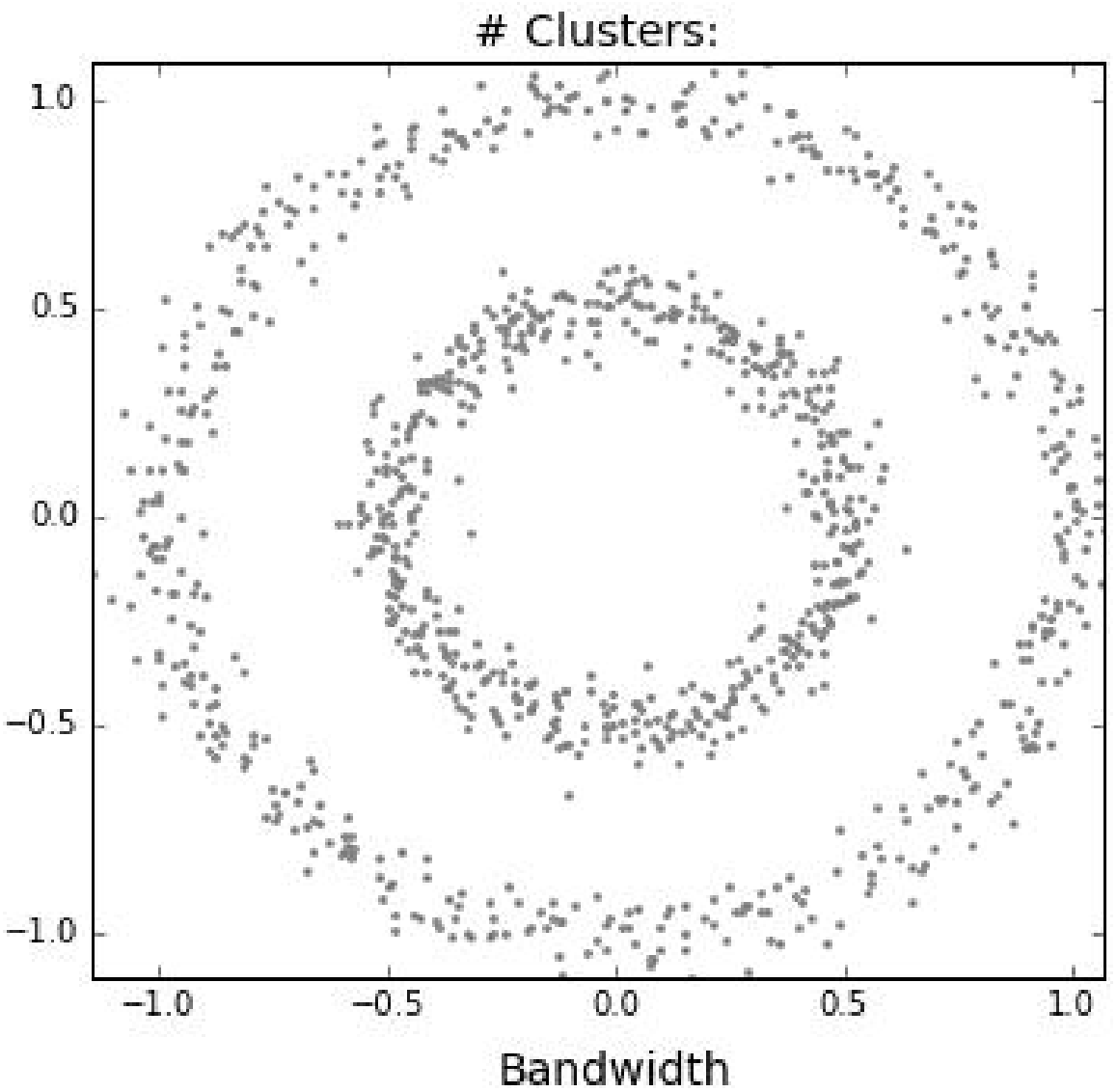
Huge bandwidth → one cluster containi



https://spin.atomicobject.com/mean-shift-clustering/

Segmentation





Segmentation

Mean-Shift is a non-parametric algorithm for partitioning multidimensional data. It is an iterative algorithm which aims to make a point converge towards the nearest local maximum.

It can be used for segmentation, by immersing the image in a 5-dimensional space, where each pixel is represented by a point having as coordinates its position in x, in y, and its R, G, B values.

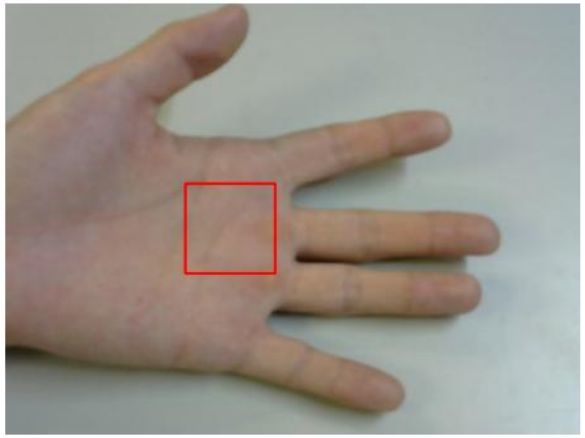
Segmentation using color

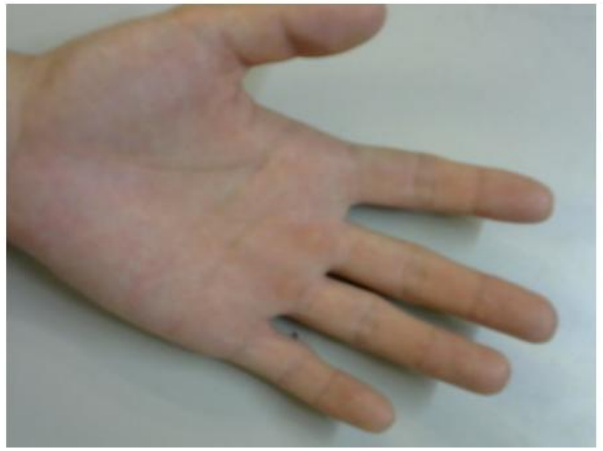
Object tracking with a Mean-Shift algorithm can be divided into three stages:

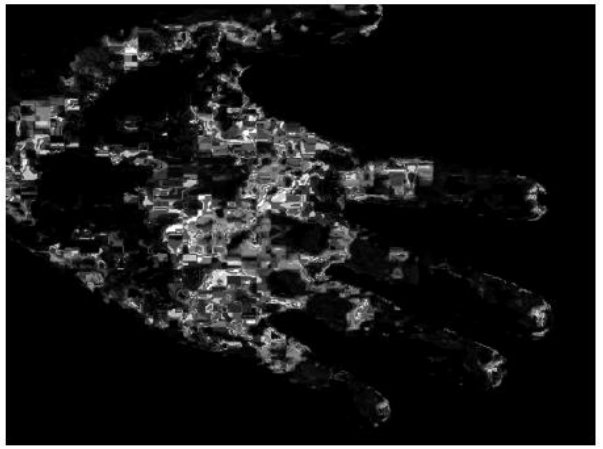
The target model is represented with a color histogram. When the object moves, this movement will be reflected in the histogram.

Create a confidence map in the new image based on the color histogram of the object in the previous image, and use mean shift to find the peak of a confidence map near the object's old position.

The confidence map is a probability density function on the new image, assigning each pixel of the new image a probability, which is the probability of the pixel color occurring in the object in the previous image.







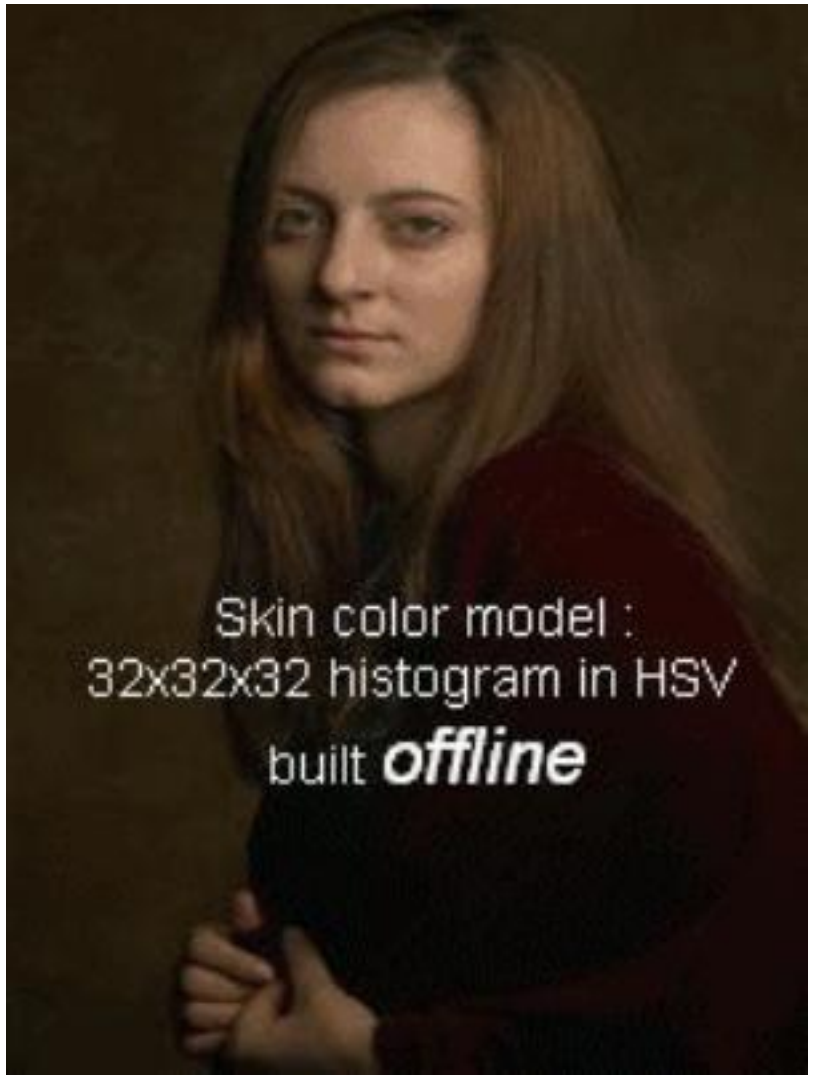
Segmentation using color

Object tracking with a Mean-Shift algorithm can be divided into three stages:

The target model is represented with a color histogram. When the object moves, this movement will be reflected in the histogram.

Create a confidence map in the new image based on the color histogram of the object in the previous image, and use mean shift to find the peak of a confidence map near the object's old position.

The confidence map is a probability density function on the new image, assigning each pixel of the new image a probability, which is the probability of the pixel color occurring in the object in the previous image.



Segmentation using color

Object tracking with a Mean-Shift algorithm can be divided into three stages:

- Target the object : Choose in the first frame the initial location of the object to be tracked.

The target model is represented with a color histogram. When the object moves, this movement will be reflected in the histogram.

- Finding the new location : In the next frame, the Mean-Shift algorithm moves the window to the new location with maximum pixel density.

The current histogram is used to search for the best target match candidate by maximizing the similarity function.

-Updating the location : The histogram and the location of the target object. are updated

Camshift (Continuously Adaptive Mean Shift) : Once meanshift converges, the Camshift algorithm updates the size of the window such that the tracking window may change in size or even rotate to better correlate to the movements of the tracked object.

Faculty of Electrical Engineering, USTHB [akourgli@usthb.dz
http://perso.usthb.dz/~akourgli/

Segmentation using color

Object tracking with a **Cam-Shift** algorithm can be divided into three stages:

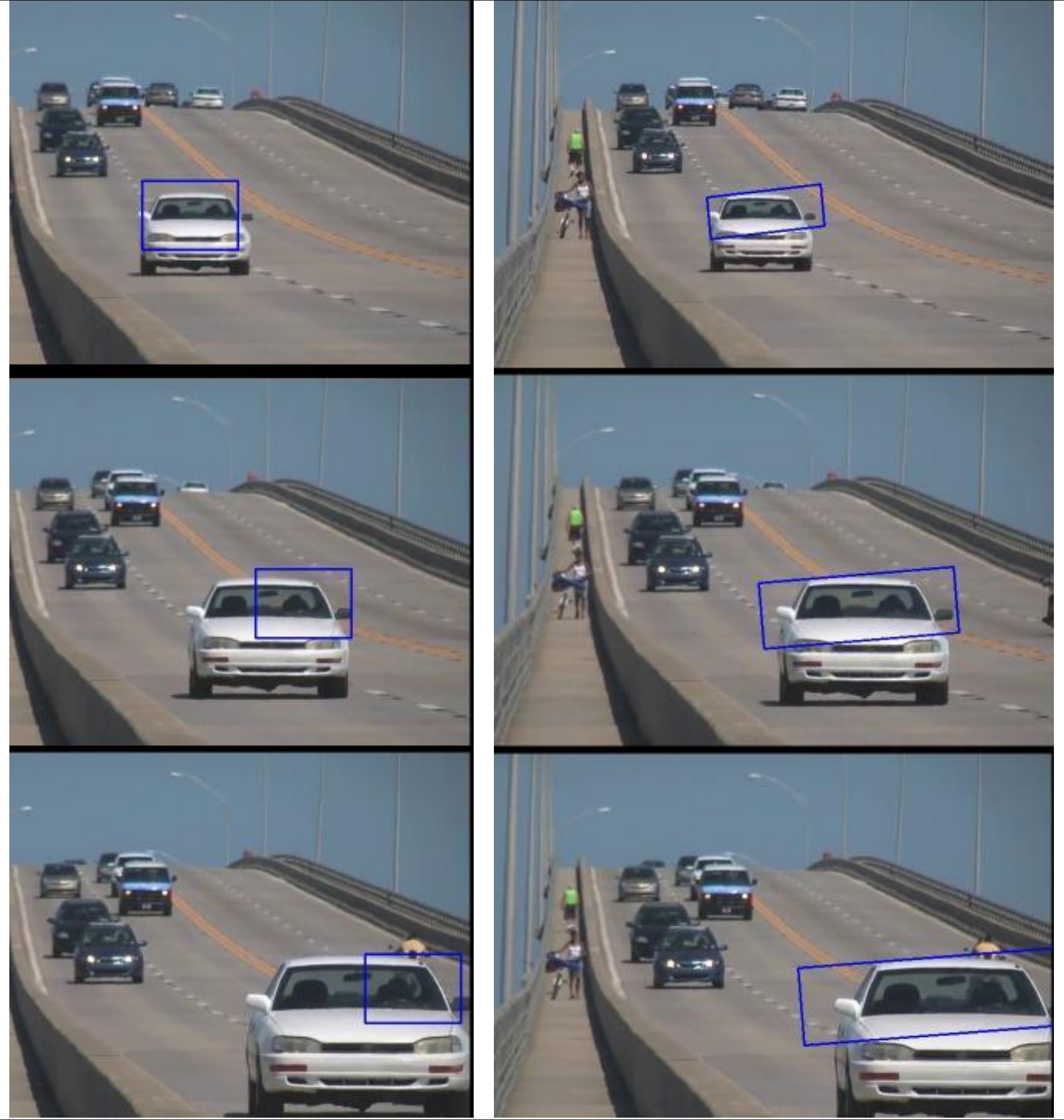
- Target the object : Choose in the first frame the initial location of the object to be tracked.

The target model is represented with a color histogram. When the object moves, this movement will be reflected in the histogram.

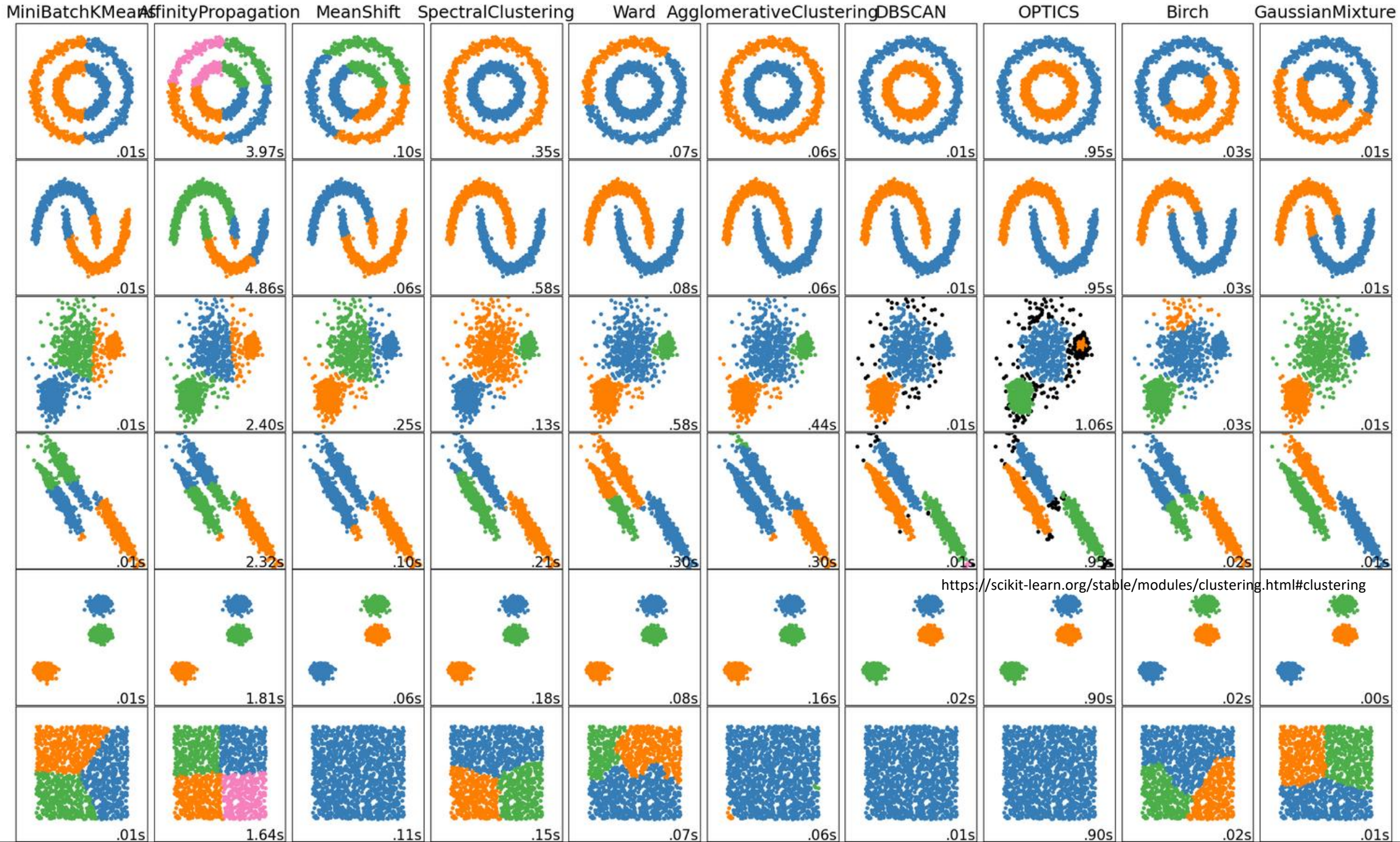
- Finding the new location : In the next frame, the Mean-Shift algorithm moves the window to the new location with maximum pixel density.

The current histogram is used to search for the best target match candidate by maximizing the similarity function.

-Updating the location : The histogram and the location of the target object are updated



Faculty of Electrical Engineering, USTHB [akourgli@usthb.dz
http://perso.usthb.dz/~akourgli/

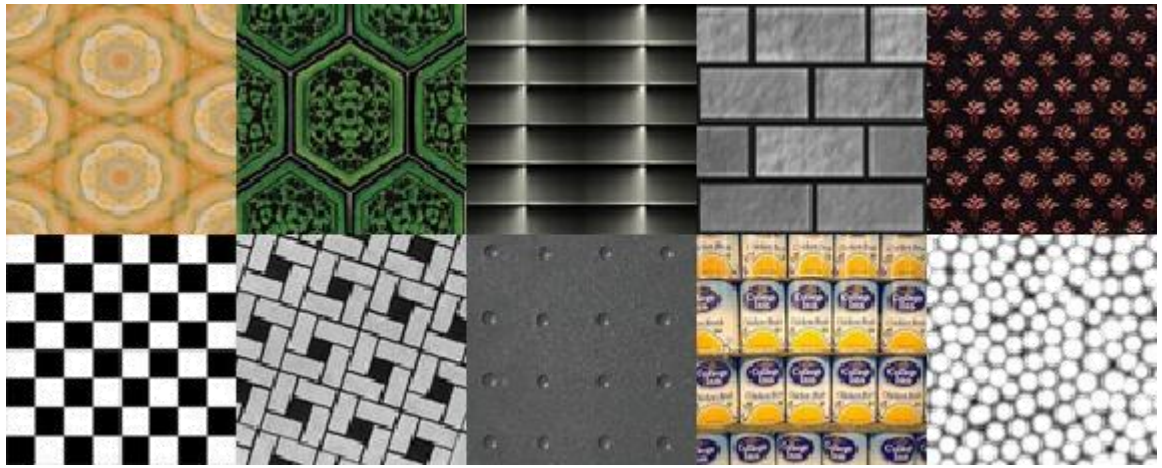


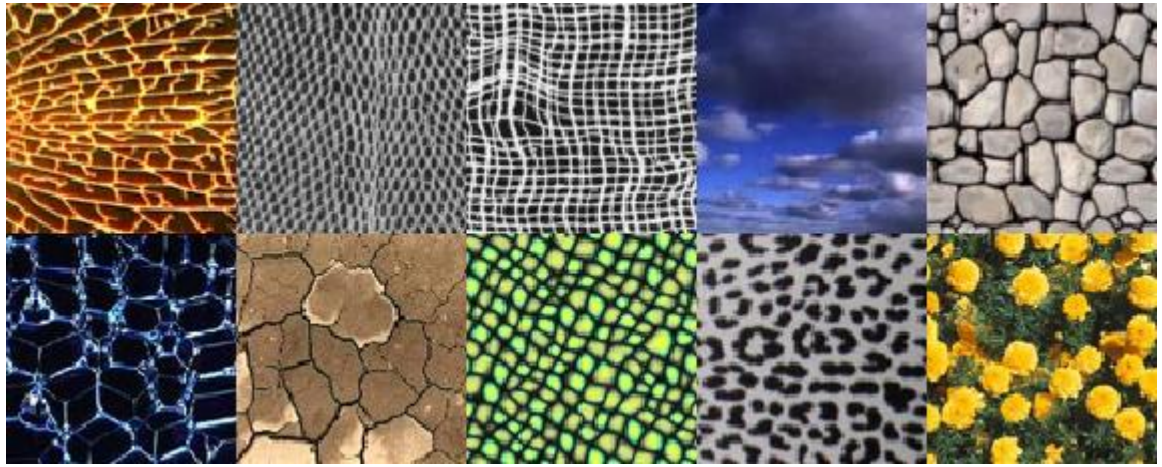
https://scikit-learn.org/stable/modules/clustering.html#clustering

Texture and Shape descriptors

Texture provides a rich source of information about the natural scene. It can be defined as a function of spatial variation of the colors (intensity) of the pixels in a neighborhood.

The surface of any visible object is textured at certain scale.





http://perso.usthb.dz/~akourgli/

https://www.cs.auckland.ac.nz/~georgy/research/texture/thesis-html/node5.html

Texture and Shape descriptors

Texture provides a rich source of information about the natural scene. It can be defined as a function of **spatial** variation of the colors (intensity) of the pixels in a **neighborhood**.

The surface of any visible object is textured at certain scale.

Images having the same histogram

Texture and Shape descriptors

For designers, a texture adds richness to a design.

As technology advances, so too do the capabilities of texturing in creating the characters, creatures and landscapes of the same video games we all know and love.

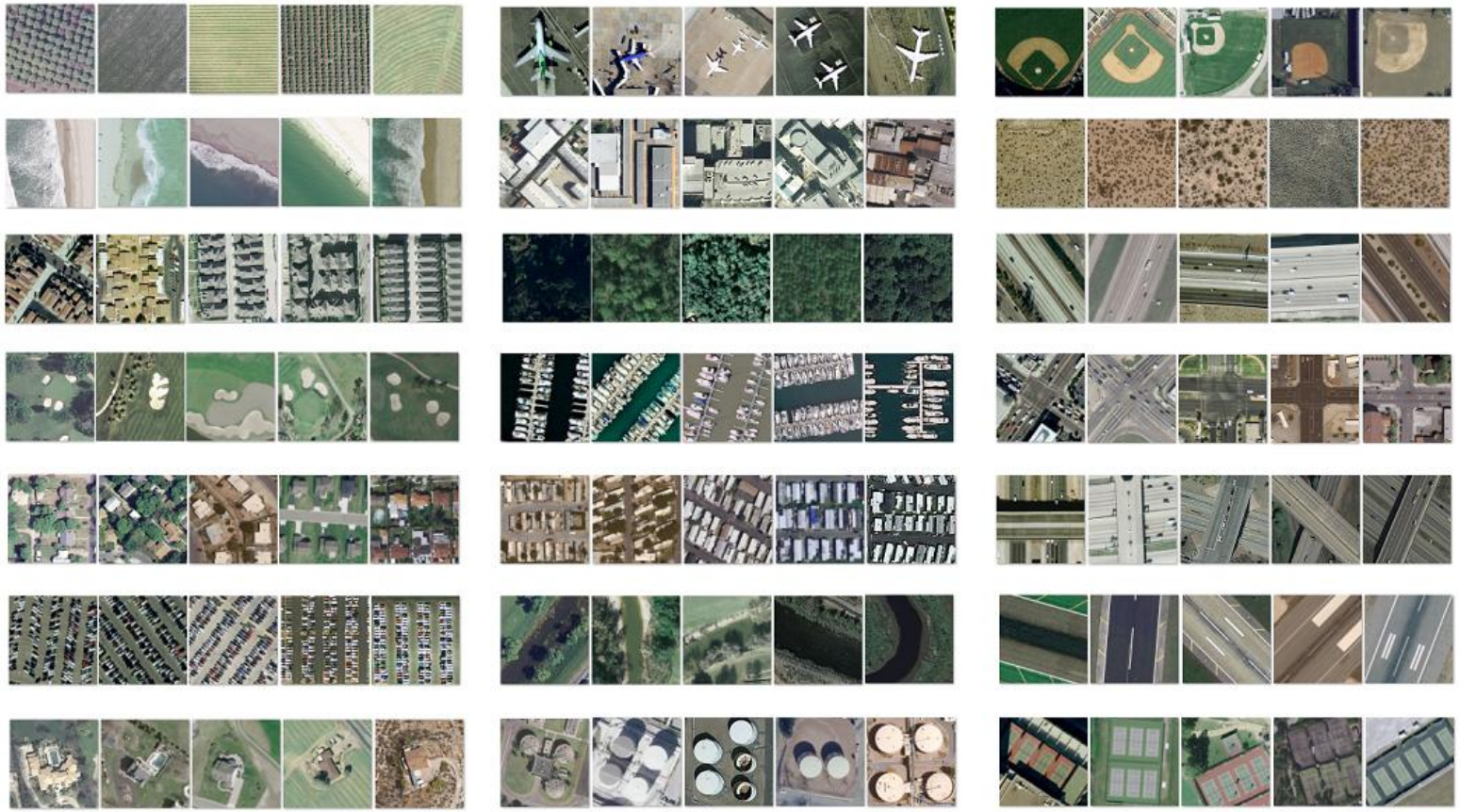
Faculty of Electrical Engineering, USTHB [akourgli@usthb.dz
http://perso.usthb.dz/~akourgli/

Texture and Shape descriptors

A texture descriptor gives us information about the spatial arrangement of color or intensities in a textured image or selected region of an image

Application of Texture Analysis

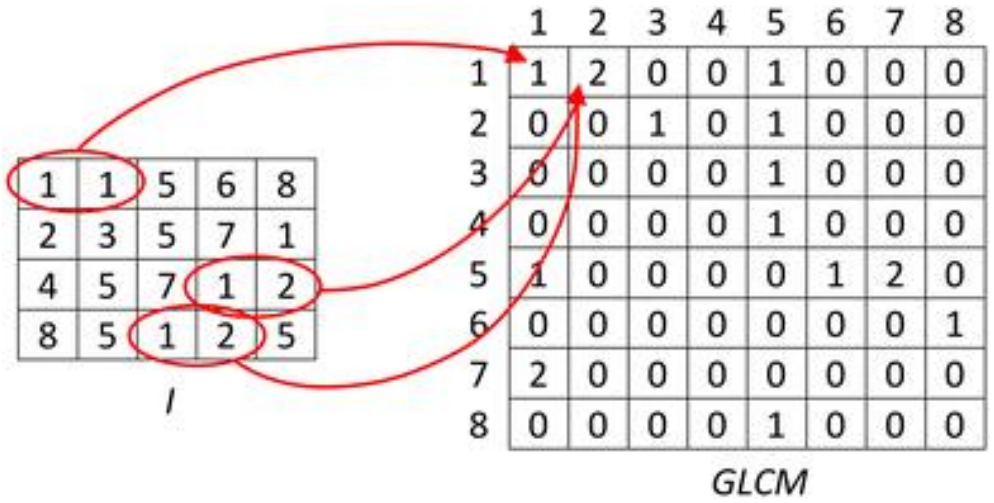
- ✓ Face detection
- ✓ Tracking objects in the videos
- ✓ Diagnosis of product quality
- ✓ Medical image analysis
- ✓ Remote sensing
- ✓ Vegetation



Texture and Shape descriptors

Gray Level Co-occurrence matrix (GLCM) : co-occurrence distribution

looks for pairs of adjacent pixel values that occur in an image and keeps recording it over the entire image.



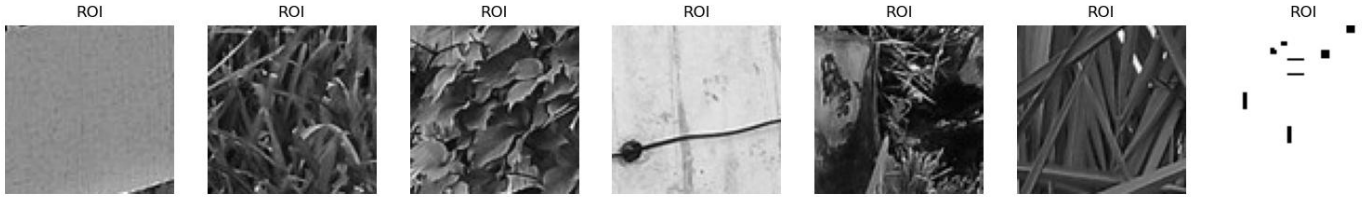
1	contrast	$f_1 = -\sum_{i=0}^{L-1}\sum_{j=0}^{L-1} \lvert i-j \rvert^2 P_{ij}$
2	entropy	$f_2 = -\sum_{i=0}^{L-1}\sum_{j=0}^{L-1} P_{ij} \log_2 P_{ij}$
3	energy	$f_5 = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} P_{ij}^2$
4	homogeneity	$f_3 = -\sum_{i=0}^{L-1}\sum_{j=0}^{L-1} \frac{P_{ij}}{1+\lvert i-j \rvert^2}$, k>=1

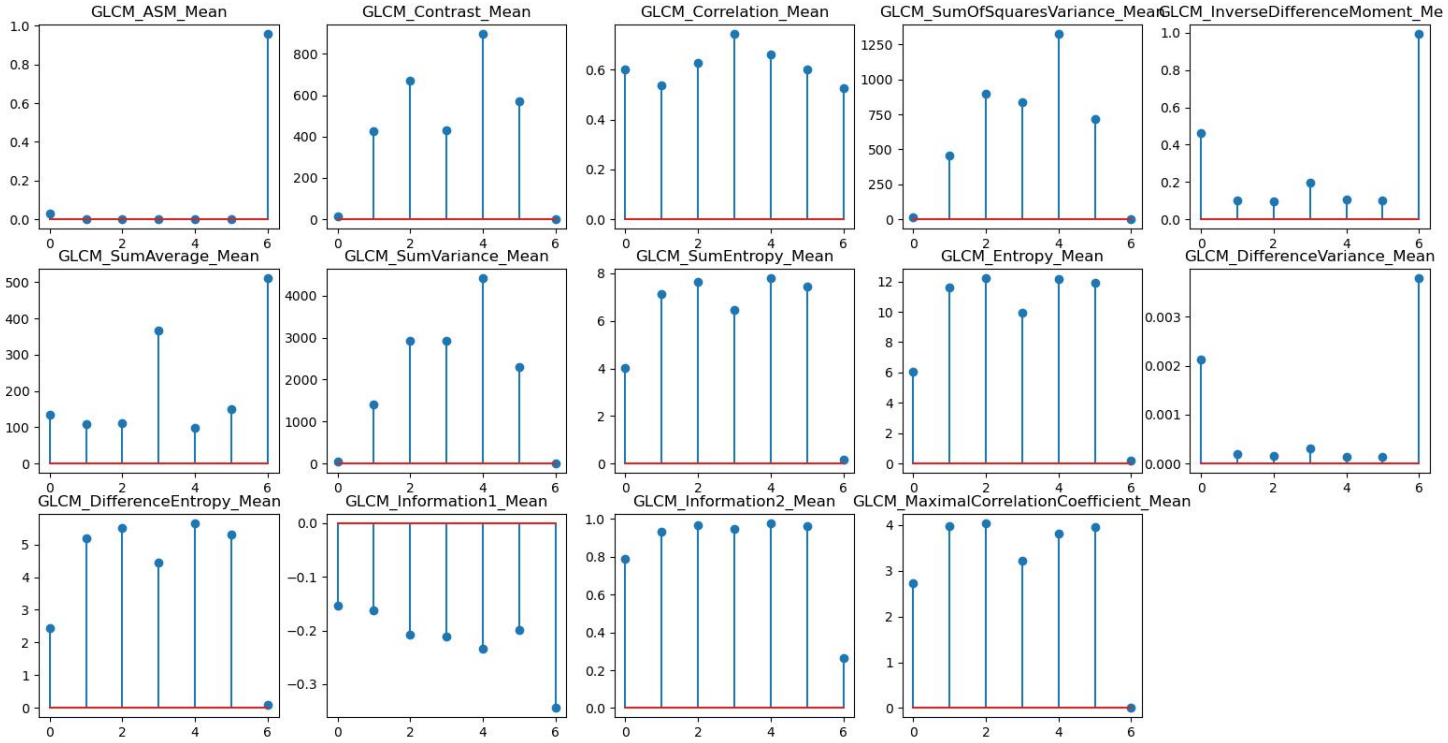
5	correlation	$f_4 = \frac{1}{\sigma_x \sigma_y}\sum_{i=0}^{L-1}\sum_{j=0}^{L-1}(i \cdot j)P_{ij} - \mu_x \mu_y$, $\mu_x = \sum_{i=0}^{L-1} i \sum_{j=0}^{L-1} P_{ij}$, $\mu_y = \sum_{j=0}^{L-1} j \sum_{i=0}^{L-1} P_{ij}$, $\sigma_x^2 = \sum_{i=0}^{L-1}(i-\mu_x)^2 \sum_{j=0}^{L-1} P_{ij}$, $\sigma_y^2 = \sum_{j=0}^{L-1}(j-\mu_y)^2 \sum_{i=0}^{L-1} P_{ij}$
6	cluster shade	$f_6 = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1}[(i-\mu_x)+(j-\mu_y)]^3 P_{ij}$
7	cluster prominence	$f_7 = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1}[(i-\mu_x)+(j-\mu_y)]^4 P_{ij}$

Texture and Shape descriptors

Gray Level Co-occurrence matrix (GLCM) : co-occurrence distribution

looks for pairs of adjacent pixel values that occur in an image and keeps recording it over the entire image.

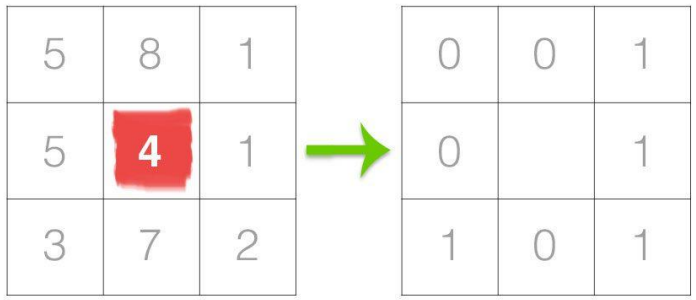


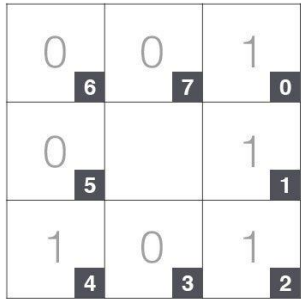


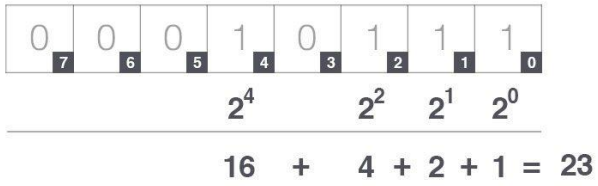
Faculty of Electrical Engineering, USTHB [akourgli@usthb.dz
http://perso.usthb.dz/~akourgli/

Texture descriptors

LBP, is a simple and grayscale invariant texture descriptor measure for classification. In LBP, a binary code is generated at each pixel by thresholding it's neighbouhood pixels to either 0 or 1 based on the value of the centre pixel.



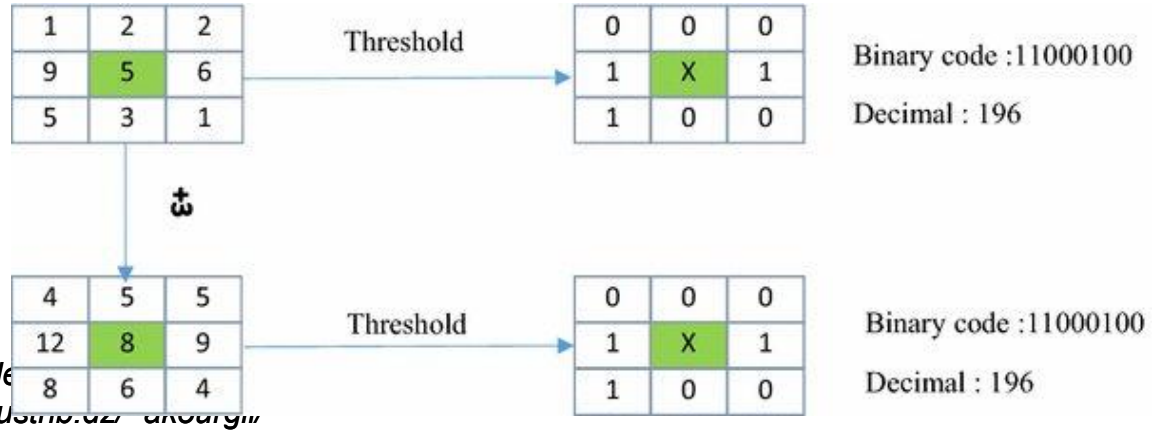


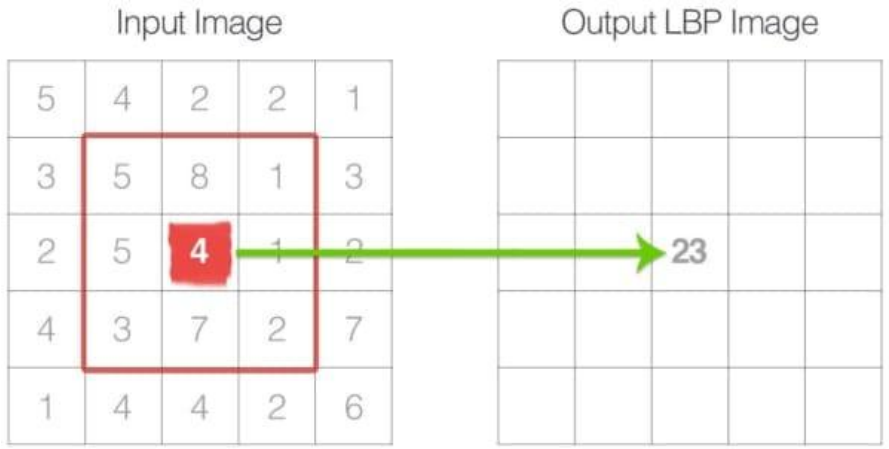


https://pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/

Collect the thresholding values either clockwise or anti-clockwise

Then, convert the binary code into decimal and place it at center of matrix.

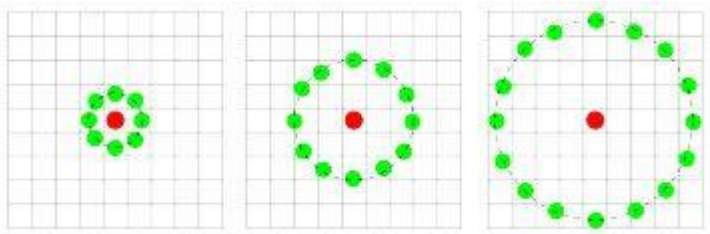




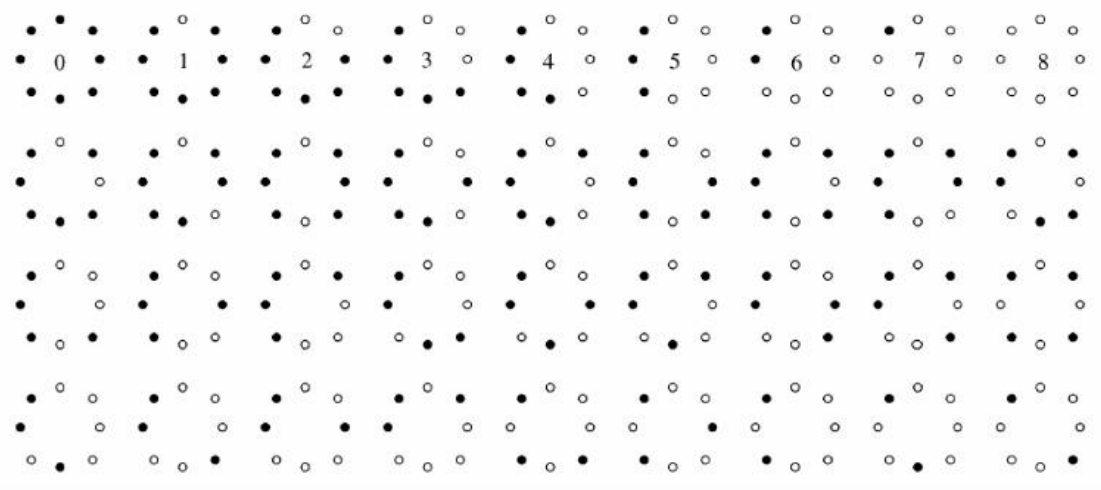
Texture descriptors

LBP Extension

- The number of points p in a circularly symmetric neighborhood
- The radius of the circle r, which allows us to account for differen



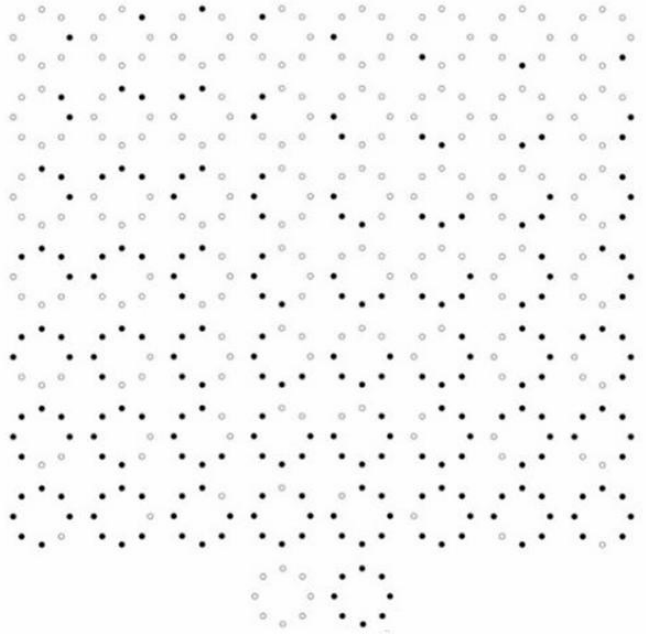
https://pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/

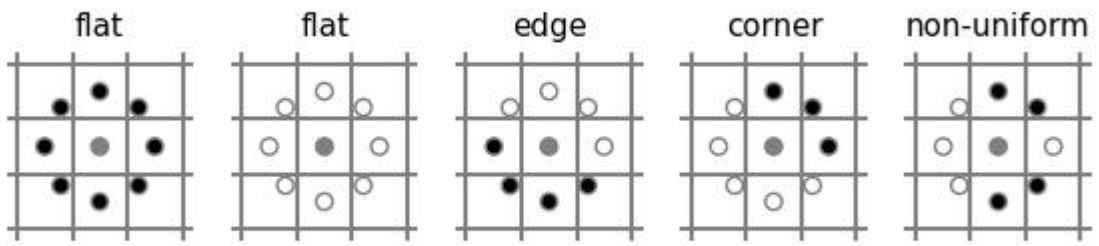


- Rotation invariant R=8 → 36 codes
- Uniform LBP uniformity : A LBP is considered to be uniform

if it has at most two 0-1 or 1-0 transitions → 59 codes

- RI Uniform LBP uniformity → 10 codes





Texture descriptors

LBP Extension

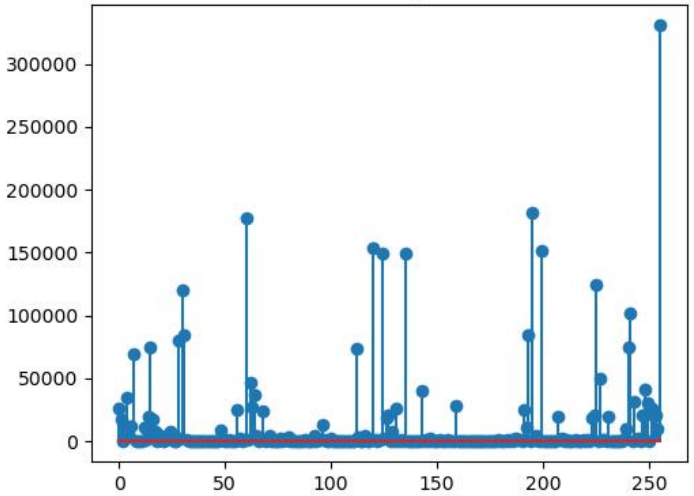
Image

LBP

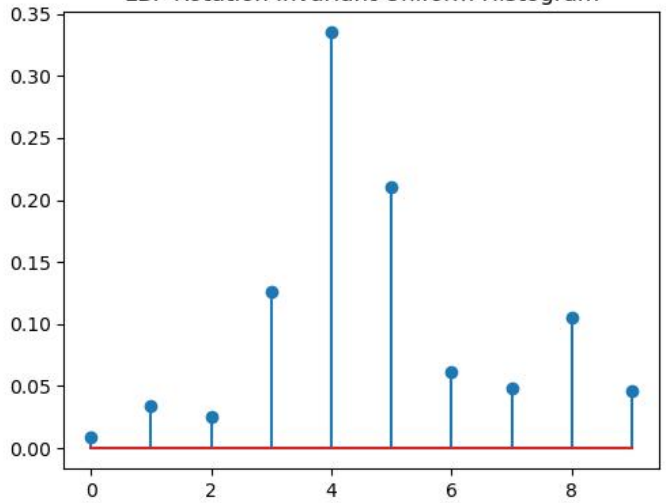
LBP Rotation Invariant

LBP uniform

LBP Histogram

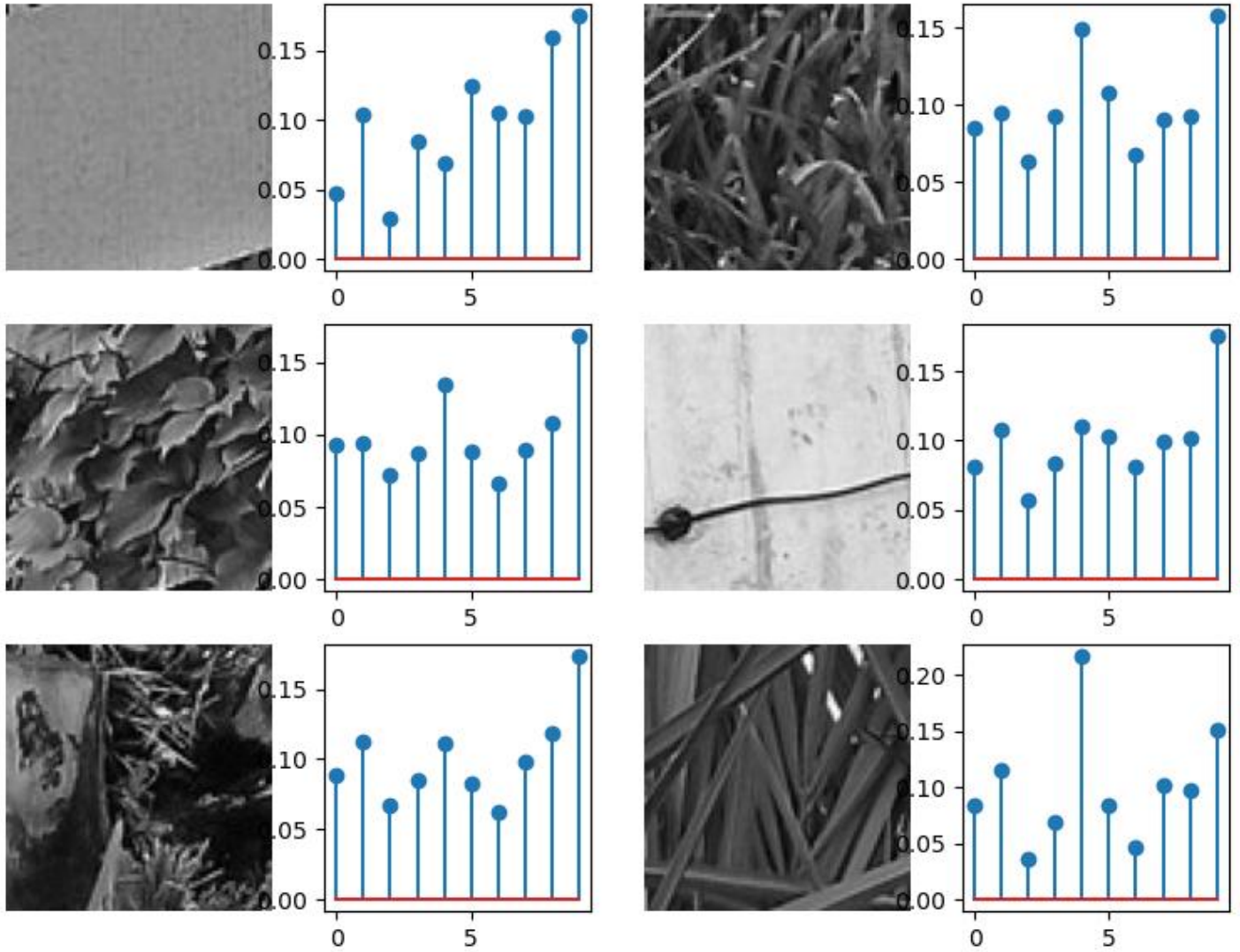


LBP Rotation Invariant Uniform Histogram



Texture descriptors

LBP Extension



Faculty of Electrical Engineering, USTHB [akourgli@usthb.dz
http://perso.usthb.dz/~akourgli/

Texture and Shape descriptors

skimage.feature.blob_dog	Finds blobs in the given grayscale image.
skimage.feature.blob_doh	Finds blobs in the given grayscale image.
skimage.feature.blob_log	Finds blobs in the given grayscale image.
skimage.feature.canny	Edge filter an image using the Canny algorithm.
skimage.feature.corner_fast	Extract FAST corners for a given image.
skimage.feature.corner_foerstner	Compute Foerstner corner measure response image.
skimage.feature.corner_harris	Compute Harris corner measure response image.
skimage.feature.corner_kitchen_rosenfeld	Compute Kitchen and Rosenfeld corner measure response image.
skimage.feature.corner_moravec	Compute Moravec corner measure response image.
skimage.feature.corner_orientations	Compute the orientation of corners.

skimage.feature.corner_peaks	Find peaks in corner measure response image.
skimage.feature.corner_shi_tomasi	Compute Shi-Tomasi (Kanade-Tomasi) corner measure response image.
skimage.feature.corner_subpix	Determine subpixel position of corners.
skimage.feature.daisy	Extract DAISY feature descriptors densely for the given image.
skimage.feature.draw_haar_like_feature	Visualization of Haar-like features.
skimage.feature.draw_multiblock_lbp	Multi-block local binary pattern visualization.
skimage.feature.fisher_vector	Compute the Fisher vector given some descriptors/ vectors, and an associated estimated GMM.
skimage.feature.graycomatrix	Calculate the gray-level co-occurrence matrix.
skimage.feature.graycoprops	Calculate texture properties of a GLCM.
skimage.feature.haar_like_feature	Compute the Haar-like features for a region of interest (ROI) of an integral image.

https://scikit-image.org/docs/stable/api/skimage.feature.html#

https://scikit-image.org/docs/dev/auto_examples/

Texture and Shape descriptors

skimage.feature.haar_like_feature_coord	Compute the coordinates of Haar-like features.
skimage.feature.hessian_matrix	Compute the Hessian matrix.
skimage.feature.hessian_matrix_det	Compute the approximate Hessian Determinant over an image.
skimage.feature.hessian_matrix_eigvals	Compute eigenvalues of Hessian matrix.
skimage.feature.hog	Extract Histogram of Oriented Gradients (HOG) for a given image.
skimage.feature.learn_gmm	Estimate a Gaussian mixture model (GMM) given a set of descriptors and number of modes (i.e. Gaussians).
skimage.feature.local_binary_pattern	Compute the local binary patterns (LBP) of an image.
skimage.feature.match_descriptors	Brute-force matching of descriptors.
skimage.feature.match_template	Match a template to a 2-D or 3-D image using normalized correlation.
skimage.feature.multiblock_lbp	Multi-block local binary pattern (MB-LBP).

skimage.feature.multiscale_basic_features	Local features for a single- or multi-channel nd image.
skimage.feature.peak_local_max	Find peaks in an image as coordinate list.
skimage.feature.plot_matches	Plot matched features.
skimage.feature.shape_index	Compute the shape index.
skimage.feature.structure_tensor	Compute structure tensor using sum of squared differences.
skimage.feature.structure_tensor_eigenvalues	Compute eigenvalues of structure tensor.
skimage.feature.BRIEF	BRIEF binary descriptor extractor.
skimage.feature.CENSURE	CENSURE keypoint detector.
skimage.feature.Cascade	Class for cascade of classifiers that is used for object detection.
skimage.feature.ORB	Oriented FAST and rotated BRIEF feature detector and binary descriptor extractor.
skimage.feature.SIFT	SIFT feature detection and descriptor extraction.

https://scikit-image.org/docs/stable/api/skimage.feature.html#

Shape descriptors

Efficient shape features must present some essential properties such as:

- ✓ Identifiability: shapes which are found perceptually similar by human have the same feature different from the others.
- ✓ Translation, rotation and scale invariance: the location, rotation and scaling changing of the shape must not affect the extracted features
- ✓ Affine invariance: the affine transform performs a linear mapping from 2D coordinates to other 2D coordinates that preserves the "straightness" and "parallelism" of lines.
- ✓ Noise resistance: features must be as robust as possible against noise, i.e., they must be the same whichever be the strength of the noise in a give range that affects the pattern.

https://machinelearningmastery.com/k-nearest-neighbors-classification-using-opencv/
https://www.kaggle.com/code/olaniyan/image-classification-using-knn
https://github.com/poojasrini/Image-classification-using-KNN/blob/main/Image_Classification.ipynb
https://medium.com/swlh/image-classification-with-k-nearest-neighbours-51b3a289280

Shape descriptors

HOG : Analyzes the distribution of edge orientations within an object to describe its shape and appearance.

- First, the gradient image in both the x and y directions is calculated.
- The image window is divided into small spatial regions called “cells.” For each cell, a local 1-D histogram of gradient or edge orientations is accumulated over all the pixels in that cell.

121	10	78	96	125
48	152	68	125	111
145	78	85	89	65
154	214	56	200	66
214	87	45	102	45

Gradient k = 2 $\rightarrow h = [-1 \;\; 0 \;\; 1]$ $\rightarrow g = [1 \;\; 0 \;\; -1]$

Change in X direction(G_x) = 78 − 89 = -11

Change in Y direction(G_y) = 68 − 56 = 8

Total Gradient Magnitude = $\sqrt{[(11)^2+(8)^2]}$ = 13.6

Φ = atan(Gy / Gx) = 36°

Magnitude		1							
Bin	0	20	40	60	80	100	120	140	160

Magnitude		13.6							
Bin	0	20	40	60	80	100	120	140	160

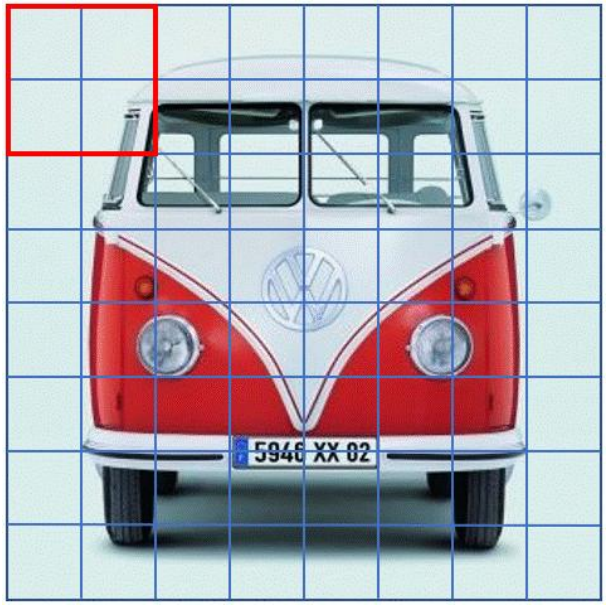
https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/

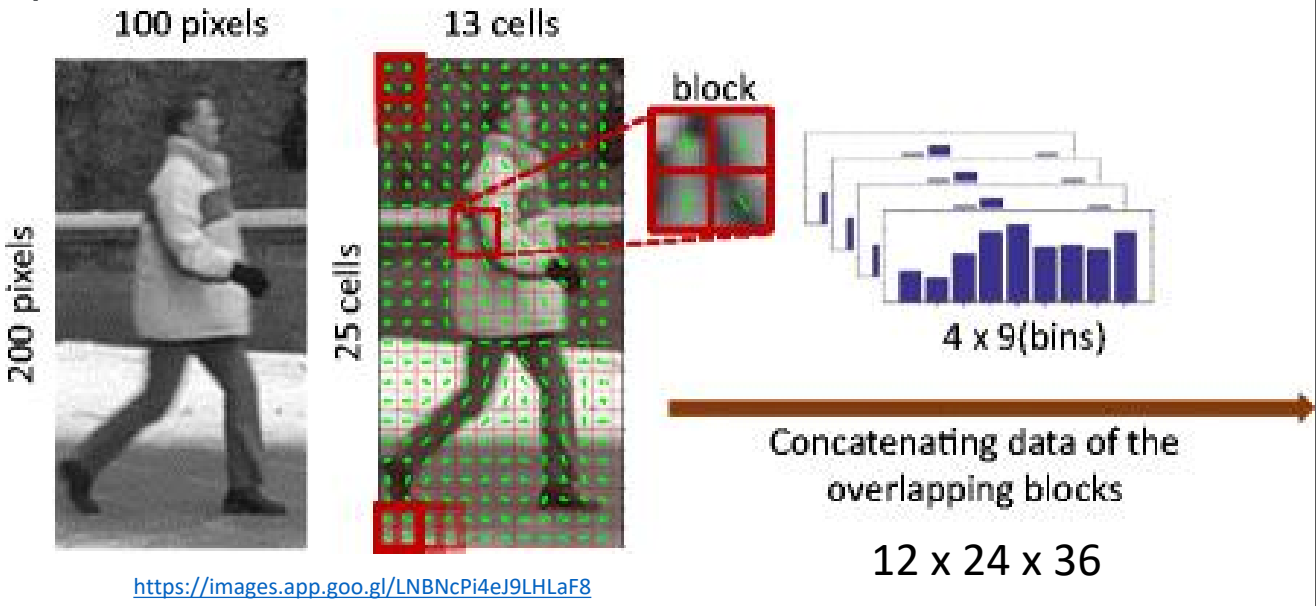
- Normalization is performed over local groups of cells (called “blocks”) to improve invariance to illumination, shadowing, and edge contrast.

Shape descriptors

HOG : Analyzes the distribution of edge orientations within an object to describe its shape and appearance.

- First, the gradient image in both the x and y directions is calculated.
- The image window is divided into small spatial regions called “cells.” For each cell, a local 1-D histogram of gradient or edge orientations is accumulated over all the pixels in that cell.





https://images.app.goo.gl/LNBNcPi4eJ9LHLaF8

- Normalization is performed over local groups of cells (called “blocks”) to improve invariance to illumination, shadowing, and edge contrast.

HOG Size = the total number of blocks x the number of cells per block x the number of orientation bins.

Shape descriptors

HOG : Analyzes the distribution of edge orientations within an object to describe its shape and appearance.

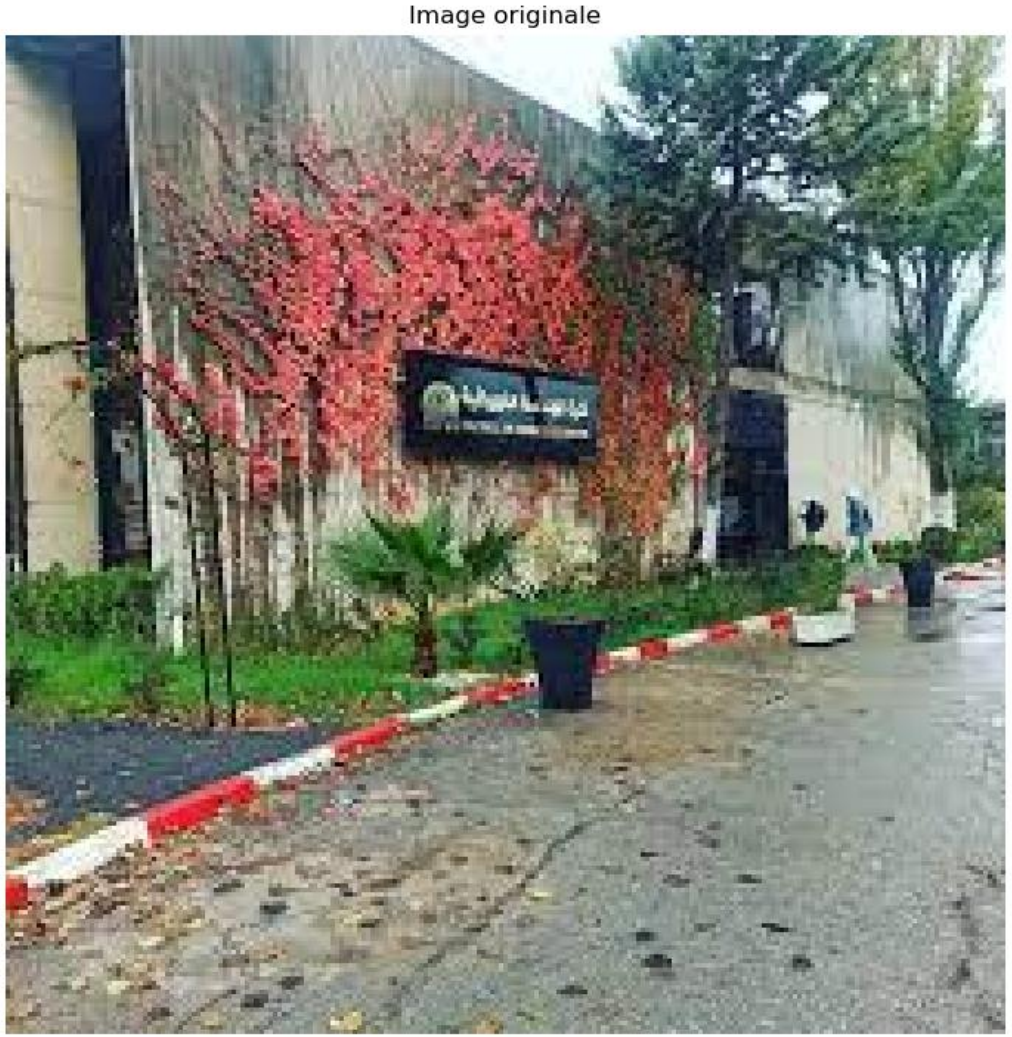
Faculty of Electrical Engineering, USTHB [akourgli@usthb.dz
http://perso.usthb.dz/~akourgli/

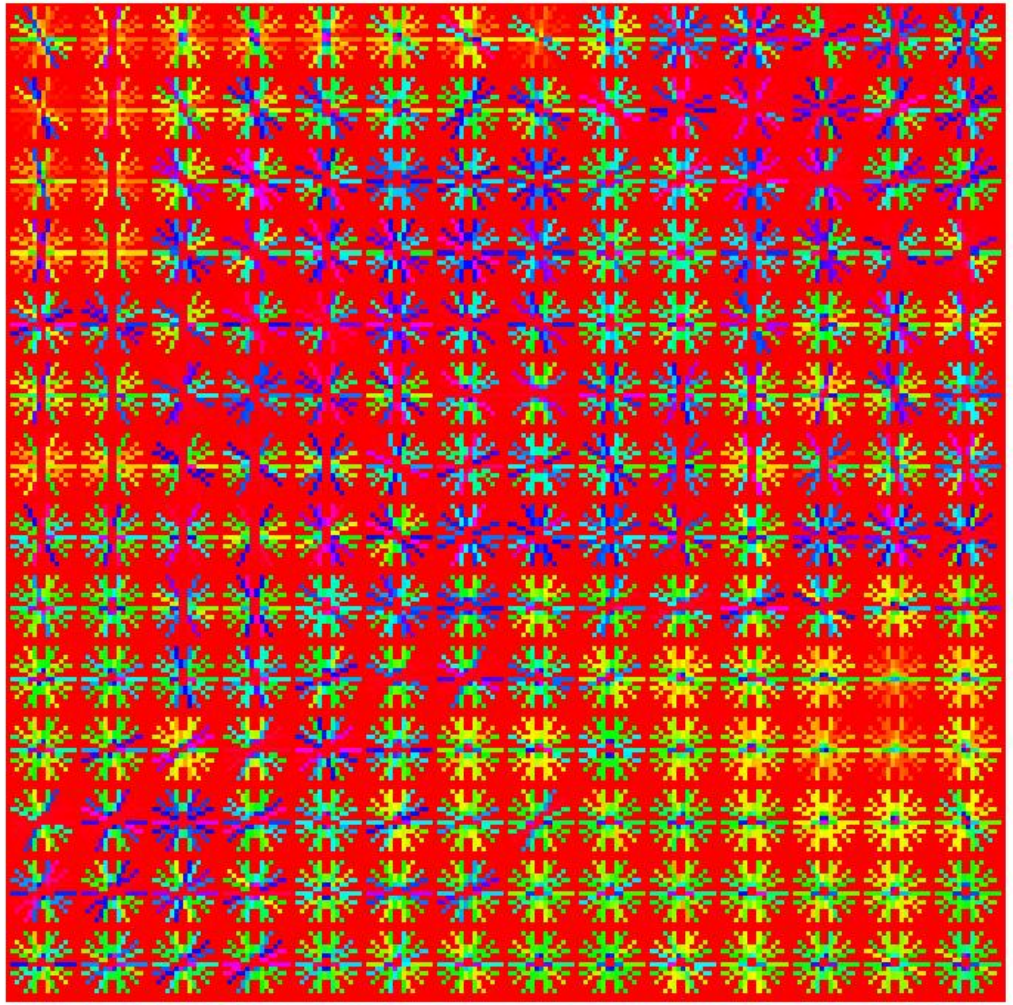
Shape descriptors

HOG : Analyzes the distribution of edge orientations within an object to describe its shape and appearance.

Shape descriptors

HOG : Analyzes the distribution of edge orientations within an object to describe its shape and appearance.

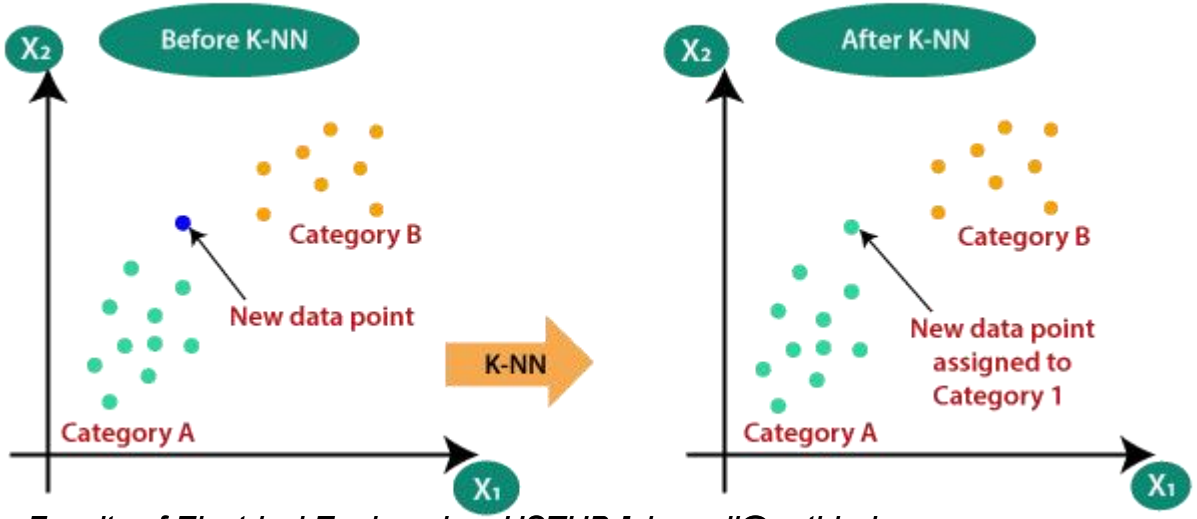


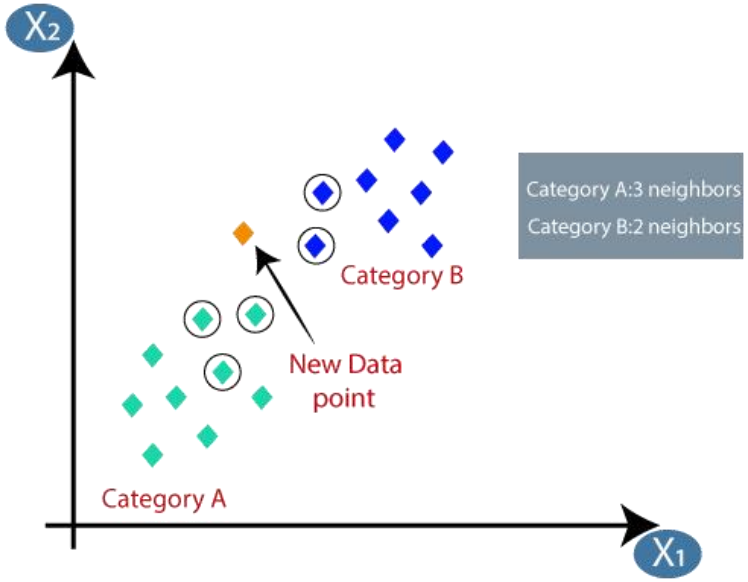


Descriptors → Classification : example K-NN classifier

K-Nearest Neighbor is one of the **simplest non parametric** Machine Learning algorithms based on Supervised Learning technique.

- It stores all the available data and classifies a new data point based on the **similarity**. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification (mainly)
- It is a **lazy learner** algorithm because it **does not learn** from the training set. At the **training phase**, it just stores the dataset and when it gets new data, then it classifies that data into a category that is **much similar** to the new data.



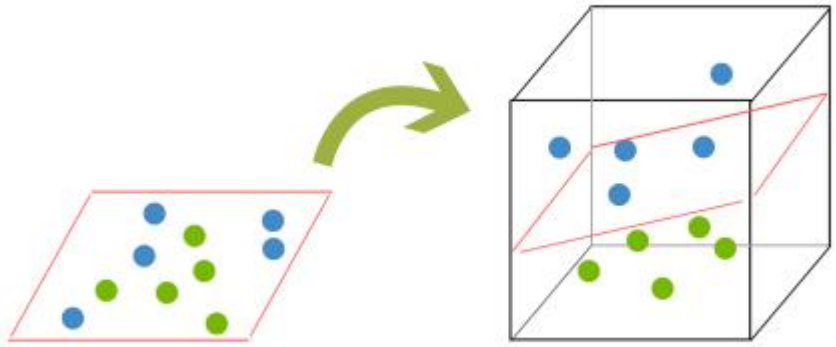


https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

Descriptors → Classification : Other classifiers

SVM is used to solve complex classification, regression, and outlier detection problems by performing optimal data transformations that **determine boundaries between data points** based on predefined classes, labels, or outputs.

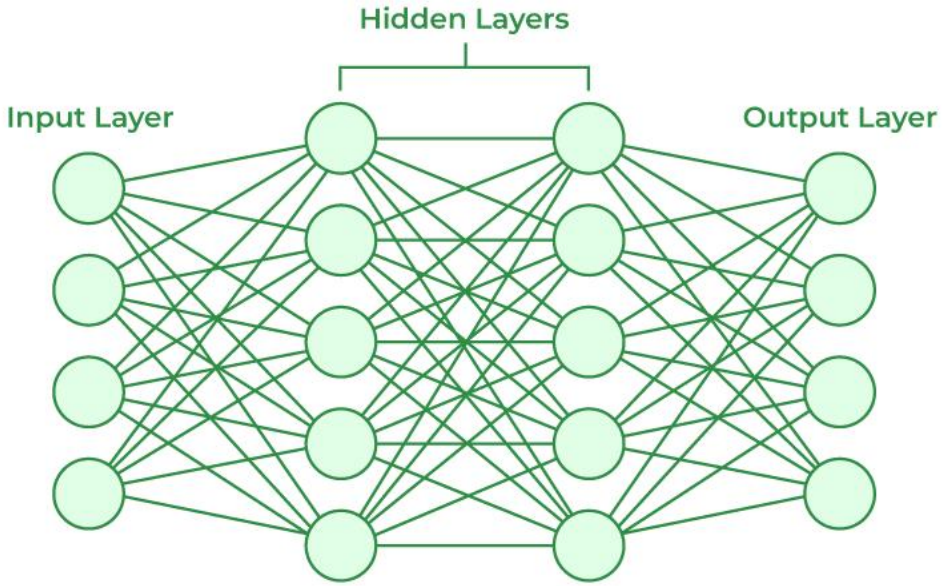
Objective : Identify a **hyperplane** that distinguishably segregates the data points of different classes in such a manner that the **largest margin** separates the classes under consideration.



https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html

NN contains nodes (**artificial neurons**) arranged in a series of **layers** interconnected from one layer to another. Each of these connections has **weights** that determine the influence of each node.

During the training phase, the NNs are trained to **minimize** the difference between the predicted output and the actual target values in a given dataset.



https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/

Descriptors → Classification : Other classifiers

1.1. Linear Models
- 1.1.1. Ordinary Least Squares
- 1.1.2. Ridge regression and classification
- 1.1.3. Lasso
- 1.1.4. Multi-task Lasso
- 1.1.5. Elastic-Net
- 1.1.6. Multi-task Elastic-Net
- 1.1.7. Least Angle Regression
- 1.1.8. LARS Lasso
- 1.1.9. Orthogonal Matching Pursuit (OMP)
- 1.1.10. Bayesian Regression
- 1.1.11. Logistic regression
- 1.1.12. Generalized Linear Models
- 1.1.13. Stochastic Gradient Descent - SGD
- 1.1.14. Perceptron
- 1.1.15. Passive Aggressive Algorithms
- 1.1.16. Robustness regression: outliers and modeling errors
- 1.1.17. Quantile Regression
- 1.1.18. Polynomial regression: extending linear models with basis functions

1.2. Linear and Quadratic Discriminant Analysis
- 1.2.1. Dimensionality reduction using Linear Discriminant Analysis
- 1.2.2. Mathematical formulation of the LDA and QDA classifiers
- 1.2.3. Mathematical formulation of LDA dimensionality reduction
- 1.2.4. Shrinkage and Covariance Estimator
- 1.2.5. Estimation algorithms

1.3. Kernel ridge regressio

1.4. Support Vector Machines
- 1.4.1. Classification
- 1.4.2. Regression
- 1.4.3. Density estimation, novelty detection
- 1.4.4. Complexity
- 1.4.5. Tips on Practical Use
- 1.4.6. Kernel functions
- 1.4.7. Mathematical formulation
- 1.4.8. Implementation details

1.5. Stochastic Gradient Descent
- 1.5.1. Classification
- 1.5.2. Regression
- 1.5.3. Online One-Class SVM
- 1.5.4. Stochastic Gradient Descent for sparse dat
- 1.5.5. Complexity
- 1.5.6. Stopping criterion
- 1.5.7. Tips on Practical Use
- 1.5.8. Mathematical formulation
- 1.5.9. Implementation details

1.6. Nearest Neighbors
- 1.6.1. Unsupervised Nearest Neighbors
- 1.6.2. Nearest Neighbors Classification
- 1.6.3. Nearest Neighbors Regression
- 1.6.4. Nearest Neighbor Algorithms
- 1.6.5. Nearest Centroid Classifier
- 1.6.6. Nearest Neighbors Transformer
- 1.6.7. Neighborhood Components Analysi

1.7. Gaussian Processes
- 1.7.1. Gaussian Process Regression (GPR)
- 1.7.2. Gaussian Process Classification (GPC)
- 1.7.3. GPC examples

1.8. Cross decomposition
- 1.8.1. PLSCanonical
- 1.8.2. PLSSVD
- 1.8.3. PLSRegression
- 1.8.4. Canonical Correlation Analysis

1.9. Naive Bayes
- 1.9.1. Gaussian Naive Bayes
- 1.9.2. Multinomial Naive Bayes
- 1.9.3. Complement Naive Bayes
- 1.9.4. Bernoulli Naive Bayes
- 1.9.5. Categorical Naive Bayes
- 1.9.6. Out-of-core naive Bayes model fitting

1.10. Decision Trees
- 1.10.1. Classification
- 1.10.2. Regression
- 1.10.3. Multi-output problems
- 1.10.4. Complexity
- 1.10.5. Tips on practical use
- 1.10.6. Tree algorithms: ID3, C4.5, C5.0 a
- 1.10.7. Mathematical formulation
- 1.10.8. Missing Values Support
- 1.10.9. Minimal Cost-Complexity Pruning

1.11. Ensembles: Gradient boosting, random forests stacking
- 1.11.1. Gradient-boosted trees
- 1.11.2. Random forests and other randomized tree
- 1.11.3. Bagging meta-estimator
- 1.11.4. Voting Classifier
- 1.11.5. Voting Regressor
- 1.11.6. Stacked generalization
- 1.11.7. AdaBoost

1.12. Multiclass and multioutput algorithms
- 1.12.1. Multiclass classification
- 1.12.2. Multilabel classification
- 1.12.3. Multiclass-multioutput classification
- 1.12.4. Multioutput regression

1.13. Feature selection
- 1.13.1. Removing features with low variance
- 1.13.2. Univariate feature selection
- 1.13.3. Recursive feature elimination
- 1.13.4. Feature selection using SelectFromModel
- 1.13.5. Sequential Feature Selection
- 1.13.6. Feature selection as part of a pipeline

1.14. Semi-supervised learning
- 1.14.1. Self Training
- 1.14.2. Label Propagation

1.15. Isotonic regression

1.16. Probability calibration
- 1.16.1. Calibration curves
- 1.16.2. Calibrating a classifier
- 1.16.3. Usage

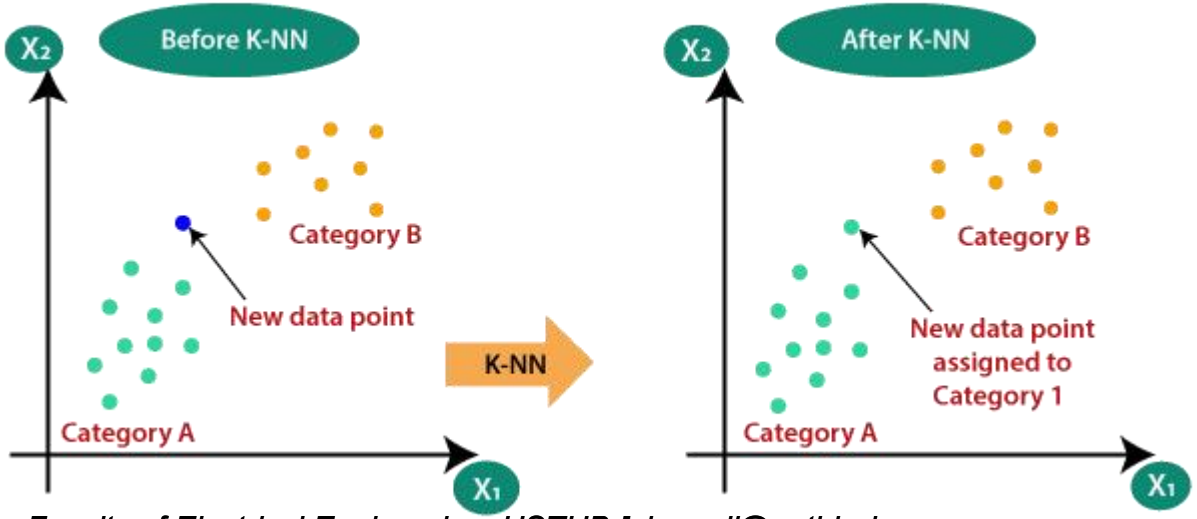
1.17. Neural network models (supervised)
- 1.17.1. Multi-layer Perceptron
- 1.17.2. Classification
- 1.17.3. Regression
- 1.17.4. Regularization
- 1.17.5. Algorithms
- 1.17.6. Complexity
- 1.17.7. Mathematical formulation
- 1.17.8. Tips on Practical Use

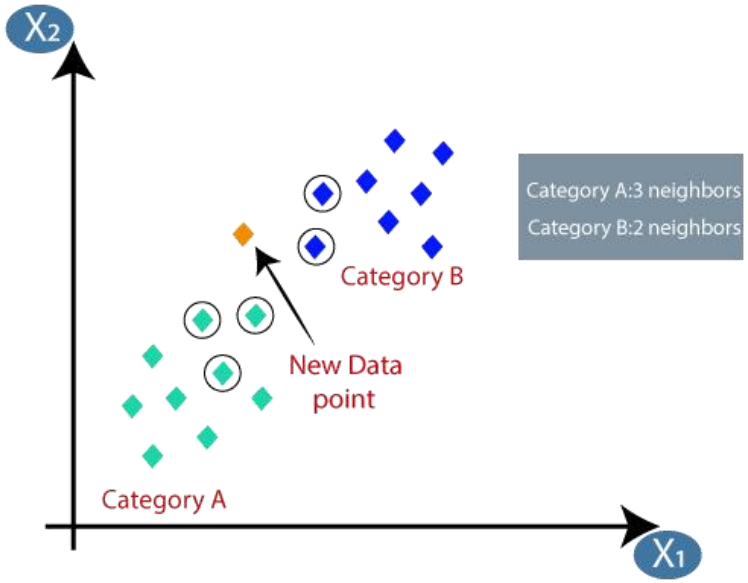
Engineering, USTHB [akourgli@usthb.dz
http://perso.usthb.dz/~akourgli/

Descriptors → Classification : example K-NN classifier

K-Nearest Neighbor is one of the **simplest non parametric** Machine Learning algorithms based on Supervised Learning technique.

- It stores all the available data and classifies a new data point based on the **similarity**. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification (mainly)
- It is a **lazy learner** algorithm because it **does not learn** from the training set. At the **training phase**, it just stores the dataset and when it gets new data, then it classifies that data into a category that is **much similar** to the new data.

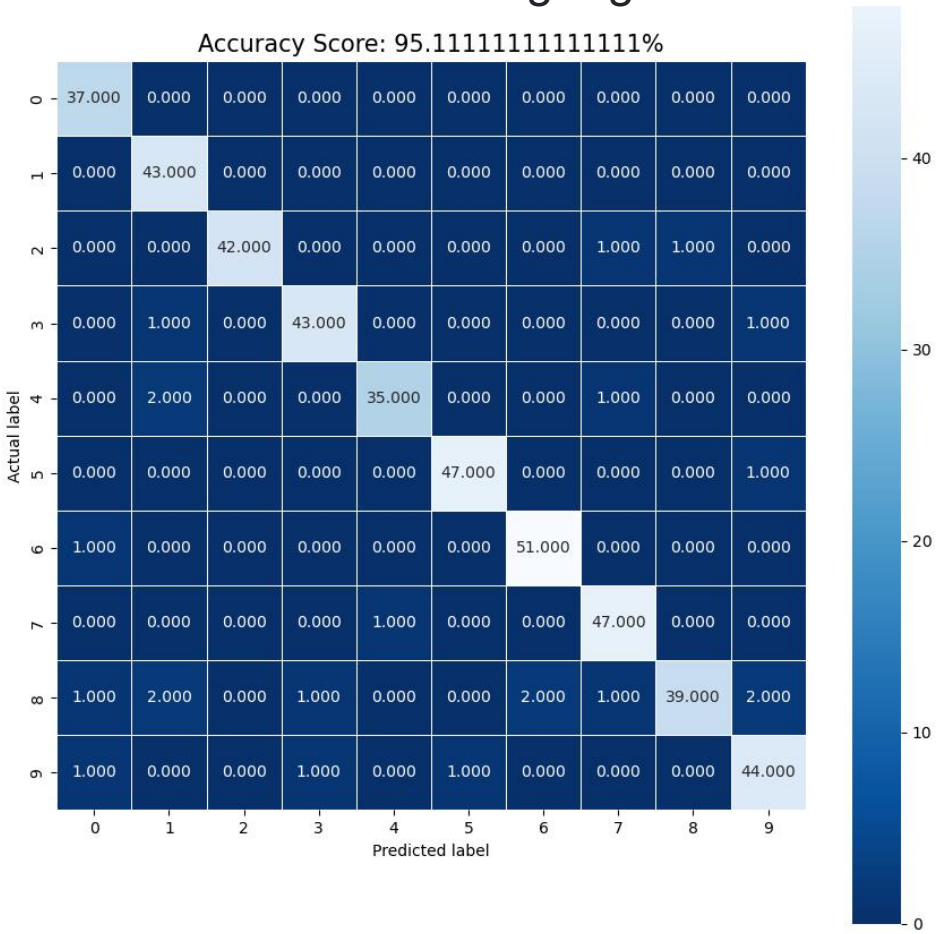




https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

Descriptors → Classification : example K-NN classifier

K-Nearest Neighbor is one of the **simplest non parametric** Machine Learning algorithms based on Supervised Learning technique.



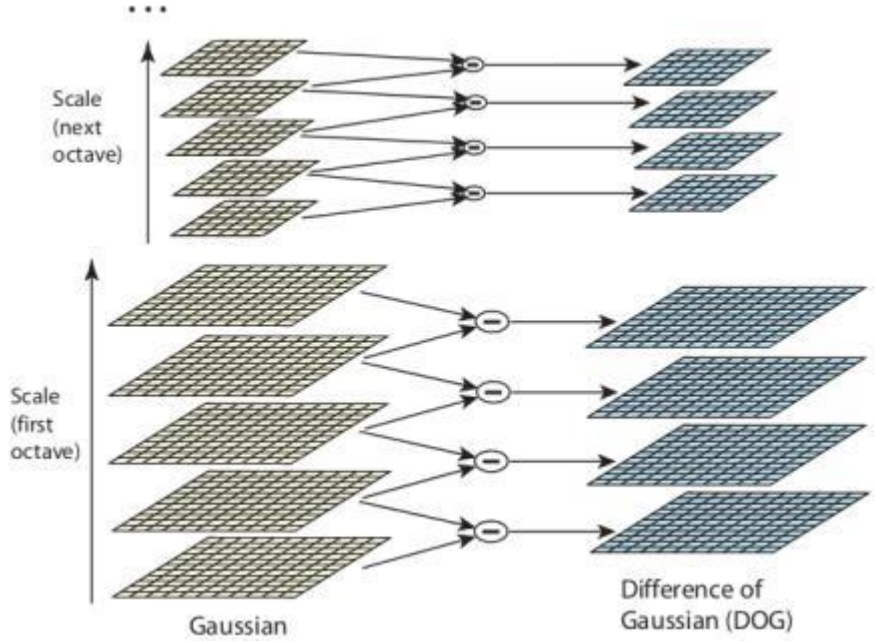
https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier
https://medium.com/swlh/histogram-of-oriented-gradients-hog-for-multiclass-image-classification-and-image-recommendation-cf0ea2caaae8
https://www.kaggle.com/code/anirbanmalick/image-classification-using-hog-knn-98-acc
https://medium.com/swlh/histogram-of-oriented-gradients-hog-for-multiclass-image-classification-and-image-recommendation-cf0ea2caaae8

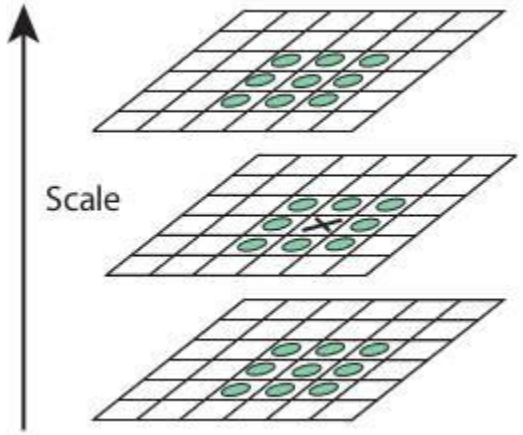
Faculty of Electrical Engineering, USTHB [akourgli@usthb.dz
http://perso.usthb.dz/~akourgli/

Shape Descriptors

SIFT (scale-invariant feature transform**)** is an algorithm that permits to detect, describe, and match local features in images**.**

1. Scale-space Extrema Detection : Laplacian of Gaussian is found for the image with various values. LoG acts as a blob detector which detects blobs in various sizes due to change in σ that acts as a scaling parameter. Once this DoG are found, images are searched for local extrema over scale and space.
2. Keypoint Localization : A concept similar to Harris corner detector is used. A 2x2 Hessian matrix (H) is used to compute the principal curvature.

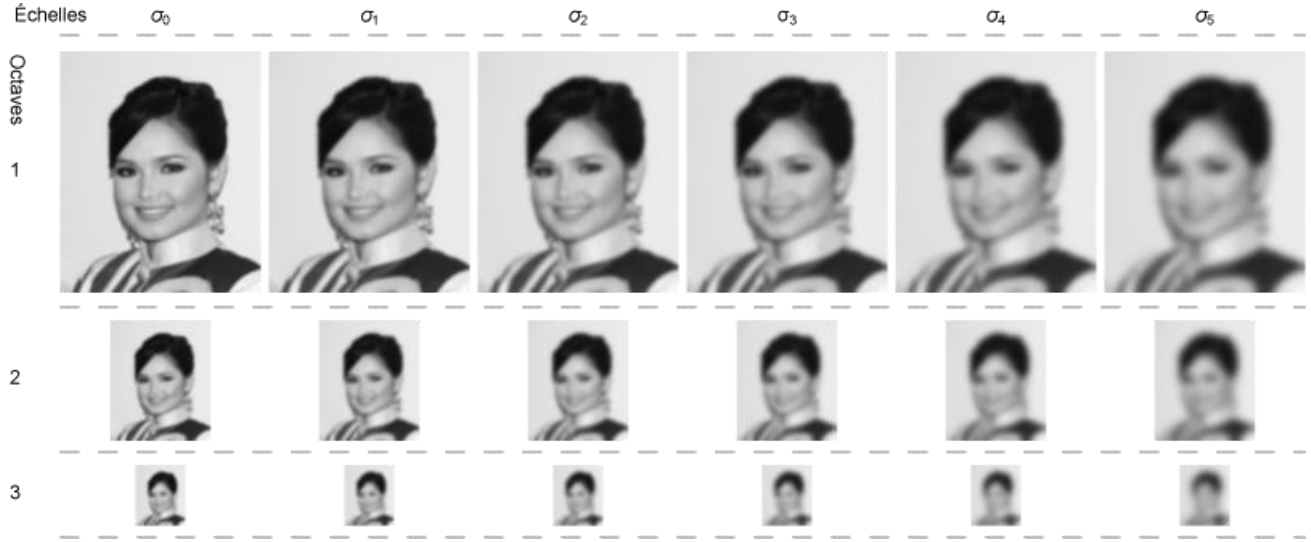




Shape Descriptors

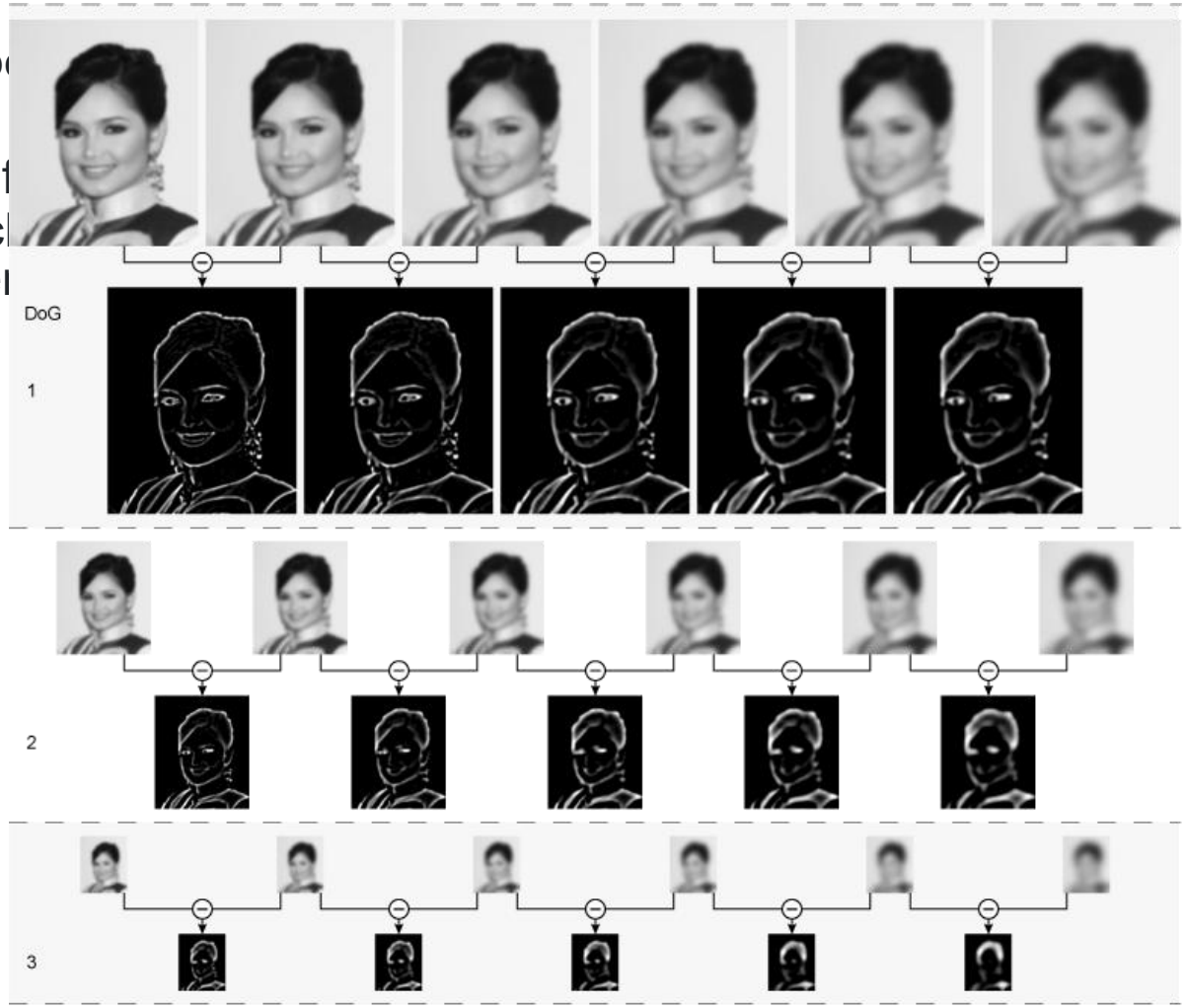
SIFT (scale-invariant feature transform**)** is an algorithm that p
images**.**

1. Scale-space Extrema Detection : Laplacian of Gaussian is f
   blob detector which detects blobs in various sizes due to c
   DoG are found, images are searched for local extrema over
2. Keypoint Localization : A concept similar to Harris corner
   compute the principal curvature.



https://www.researchgate.net/publication/331185020_A_novel_SIFT_architecture_and_ASIC_implementation_for_real_time_SOC_application/figures?lo=1

https://upload.wikimedia.org/wikipedia/commons/8/8e/Differences-of-Grandients.png

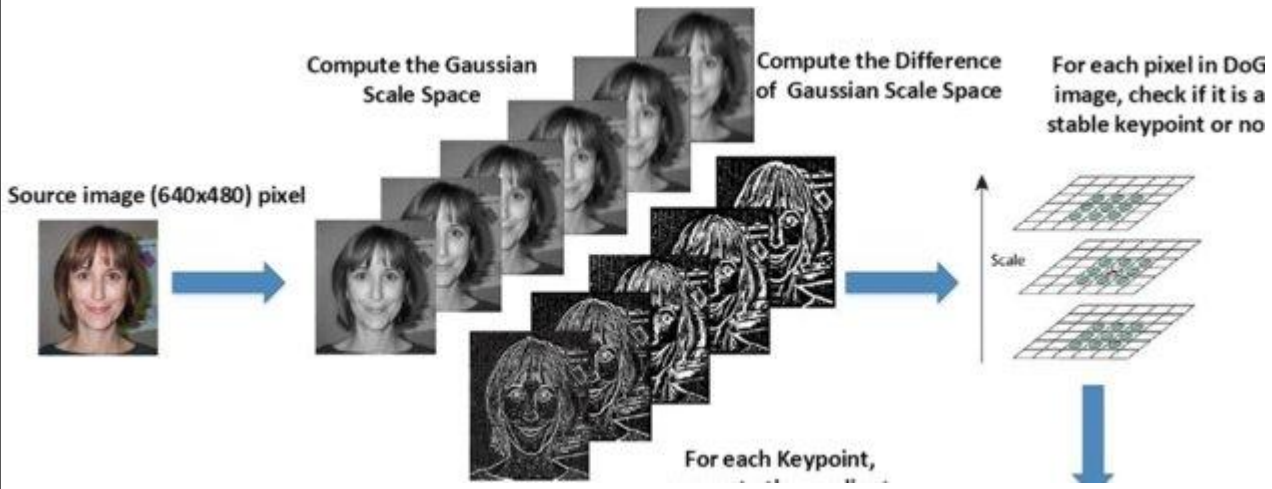


https://www.researchgate.net/publication/256546531_Workload_Analysis_and_Efficient_OpenCL-based_Implementation_of_SIFT_Algorithm_on_a_Smartphone/figures?lo=1

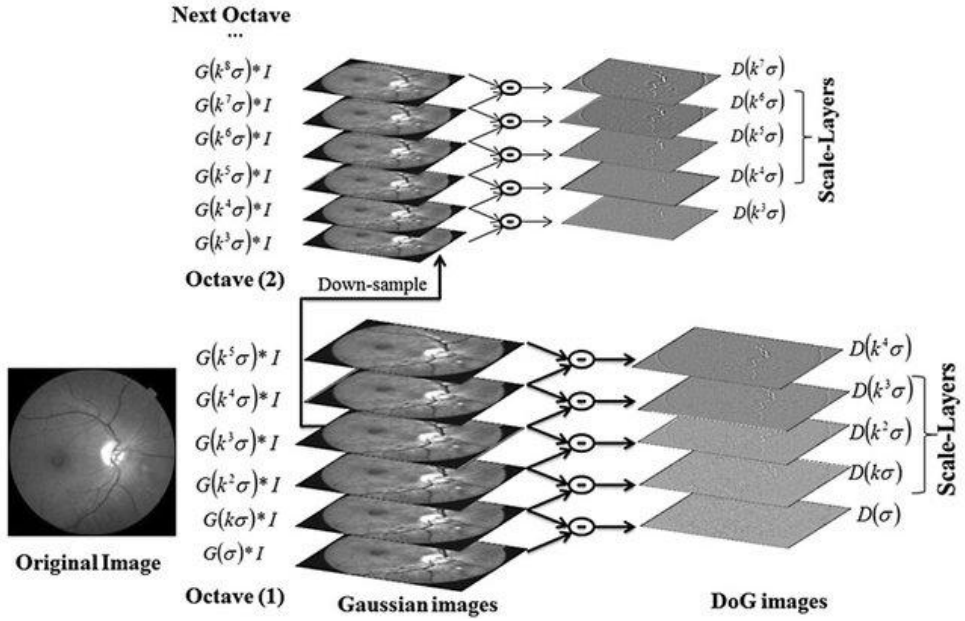
Shape Descriptors

SIFT (scale-invariant feature transform**)** is an algorithm that permits to detect, describe, and match local features in images**.**

1. Scale-space Extrema Detection : Laplacian of Gaussian is found for the image with various values. LoG acts as a blob detector which detects blobs in various sizes due to change in σ that acts as a scaling parameter. Once this DoG are found, images are searched for local extrema **over scale and space**.
2. Keypoint Localization : A concept similar to Harris corner detector is used. A 2x2 Hessian matrix (H) is used to compute the principal curvature.



https://www.researchgate.net/publication/331185020_A_novel_SIFT_architecture_and_ASIC_implementation_for_real_time_SOC_application/figures?lo=1

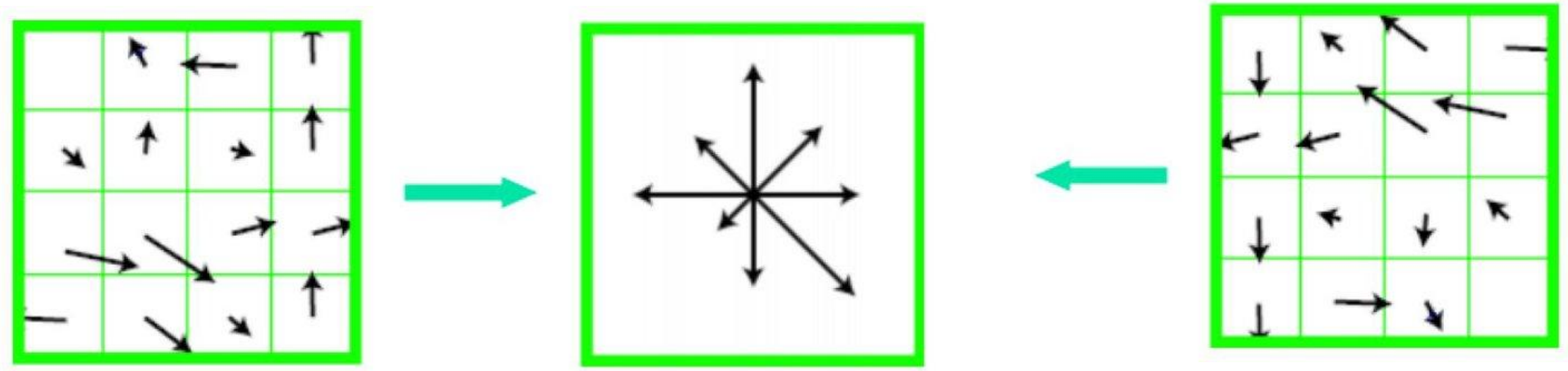


https://www.researchgate.net/publication/256546531_Workload_Analysis_and_Efficient_OpenCL-based_Implementation_of_SIFT_Algorithm_on_a_Smartphone/figures?lo=1

Shape Descriptors

SIFT (scale-invariant feature transform**)** is an algorithm that permits to detect, describe, and match local features in images**.**

3. Orientation Assignment : An orientation is assigned to each keypoint to achieve invariance to image rotation. A neighbourhood is taken around the keypoint location depending on the scale, and the gradient magnitude and direction is calculated in that region. An orientation histogram with 36 bins covering 360 degrees is created (It is weighted by gradient magnitude and gaussian-weighted circular window with σ equal to 1.5 times the scale of keypoint.

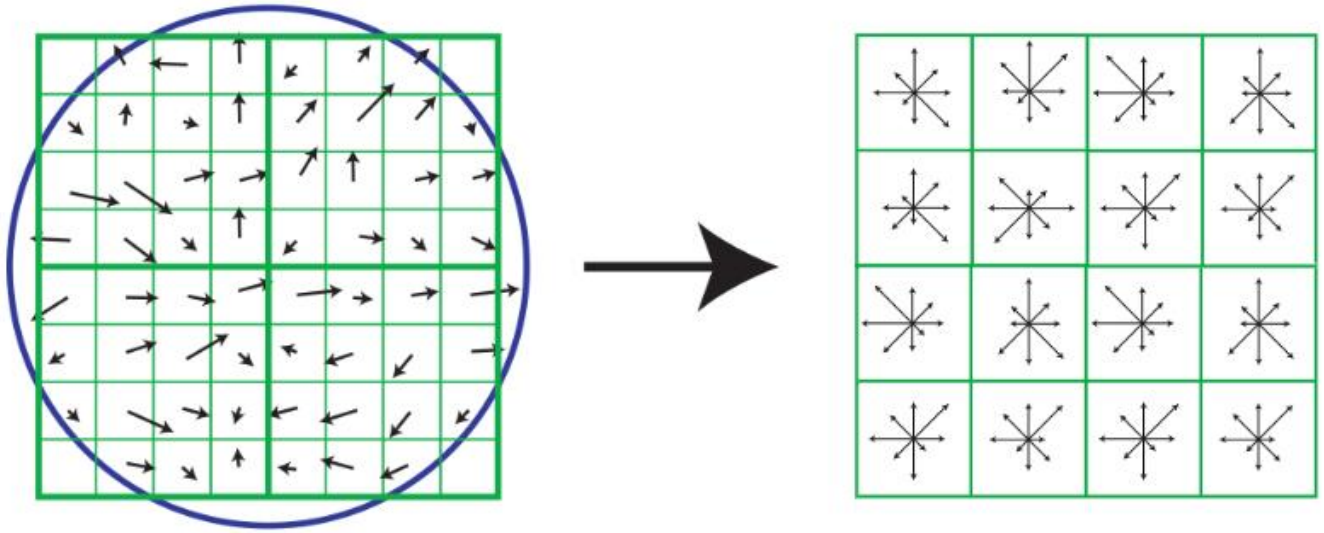


https://www.researchgate.net/figure/lassignement-de-lorientation-au-descripteur-SIFT64_fig10_344224269

Shape Descriptors

SIFT (scale-invariant feature transform**)** is an algorithm that permits to detect, describe, and match local features in images**.**

4. Keypoint Descriptor : A 16x16 neighbourhood around the keypoint is taken. It is divided into 16 sub-blocks of 4x4 size. For each sub-block, 8 bin orientation histogram is created → A total of 128 bin values are available. It is represented as a vector to form keypoint descriptor. In addition to this, several measures are taken to achieve robustness against illumination changes, rotation etc.



Shape Descriptors

SIFT (scale-invariant feature transform**)** is an algorithm that permits to detect, describe, and match local features in images**.**

Application : Robot localization, tracking; 3D modelling

See :

- **SURF** (Speeded-Up Robust Feature) improvement of SIFT utilizes Box Filters rather than using Difference of Gaussian to approximate LoG,

- ORB (Oriented FAST and Rotated BRIEF) is a fusion of FAST keypoint detector and BRIEF descriptors with many changes to enhance the performance. **FAST** (Features from Accelerated Segment Test) is used as the key-point detector while **BRIEF** (Binary Robust Independent Elementary Features) is used as the key-point descriptor

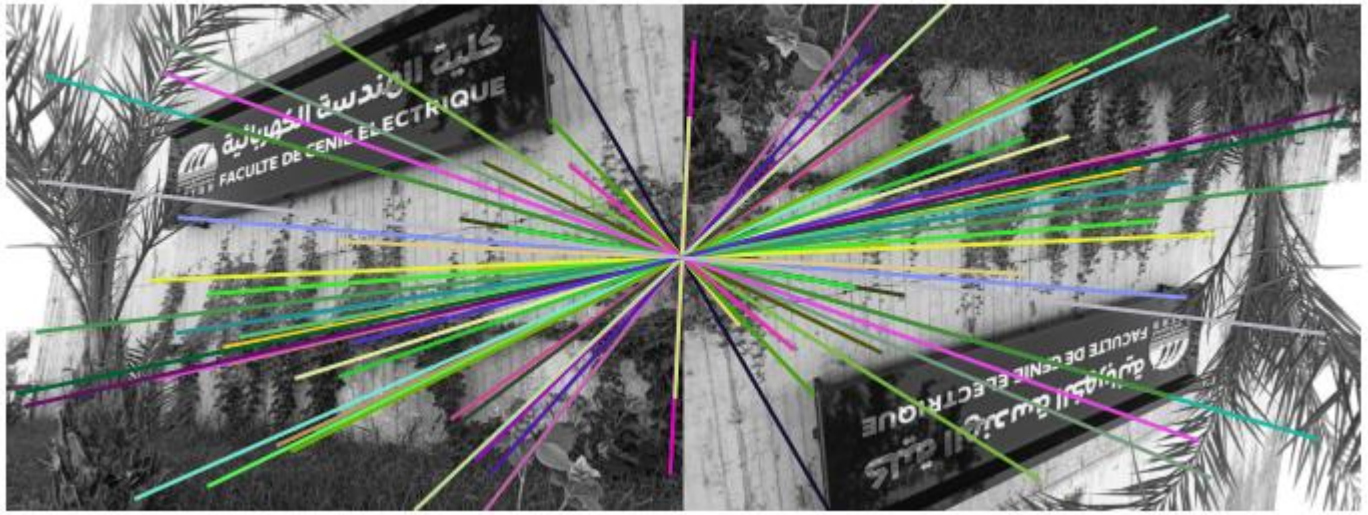
https://docs.opencv.org/3.4/db/d27/tutorial_py_table_of_contents_feature2d.html

Shape Descriptors

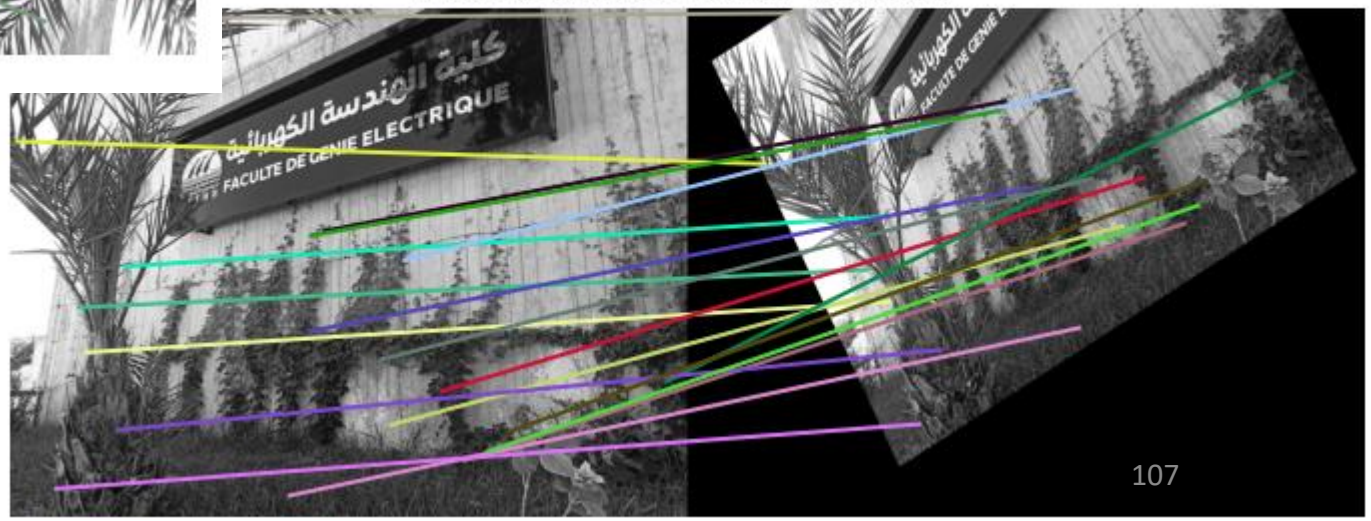
Matching using SIFT (scale-invariant feature transform**)**

.

Original Image vs. Flipped Image
(subset of matches for visibility)

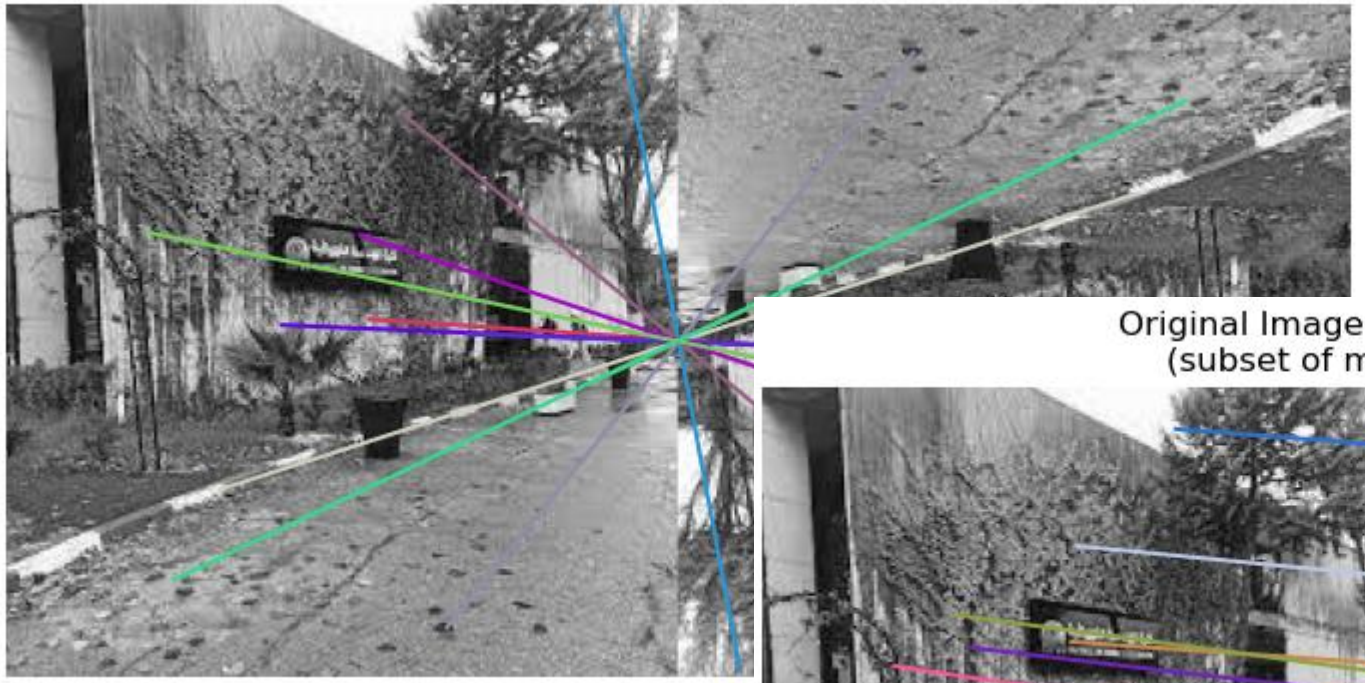


Original Image vs. Transformed Image
(subset of matches for visibility)



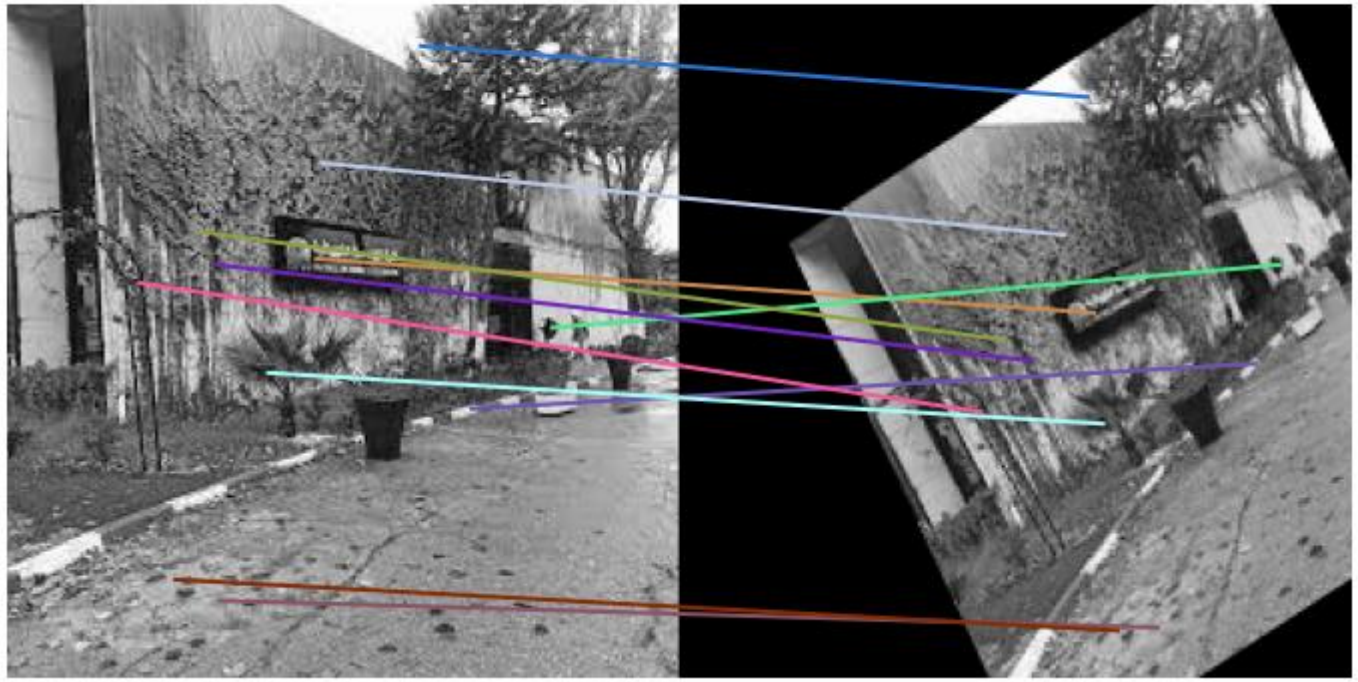
Shape Descriptors

Original Image vs. Flipped Image
(subset of matches for visibility)



mits to detect, describe, and match local features in

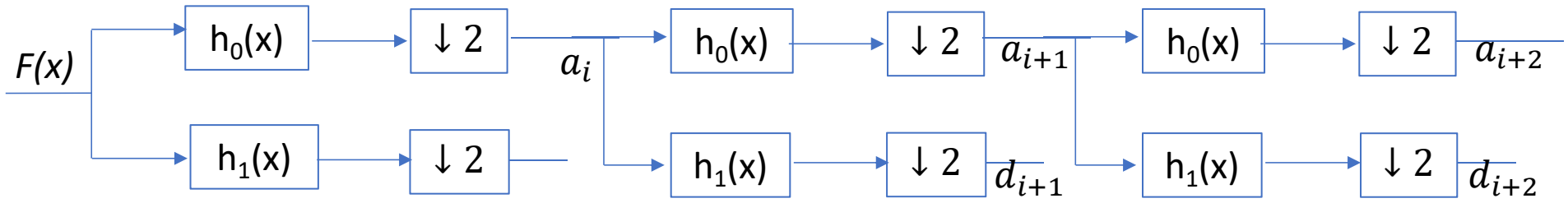
Original Image vs. Transformed Image
(subset of matches for visibility)

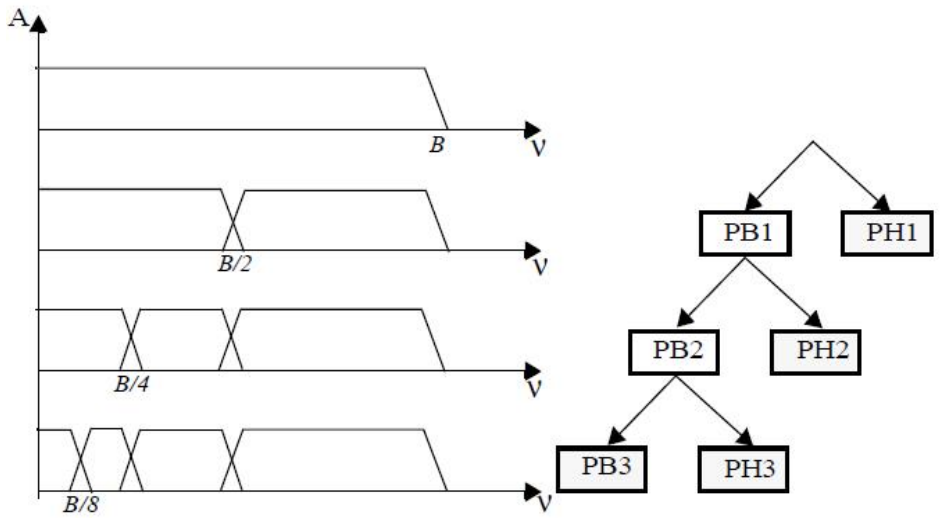


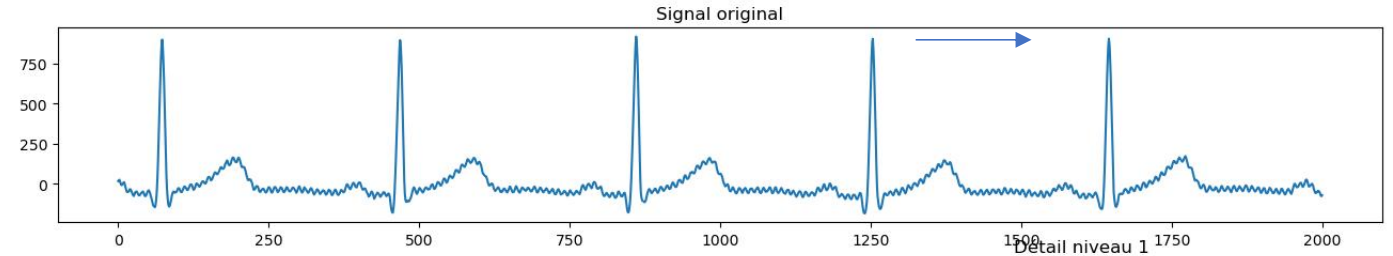
Transform Descriptors

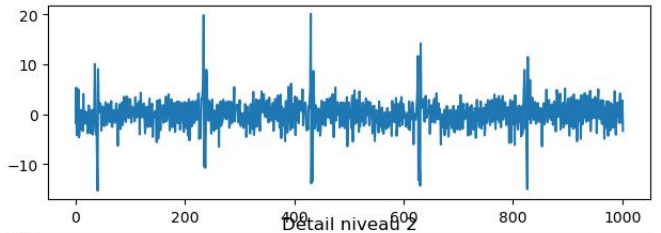
DWT → 1 dimension

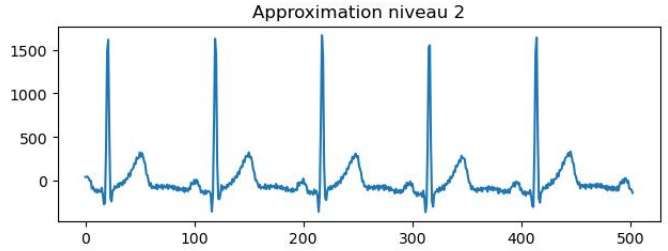
A signal is passed through two filters, high pass and low pass filters.

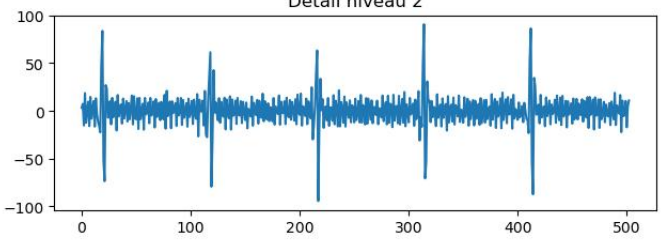










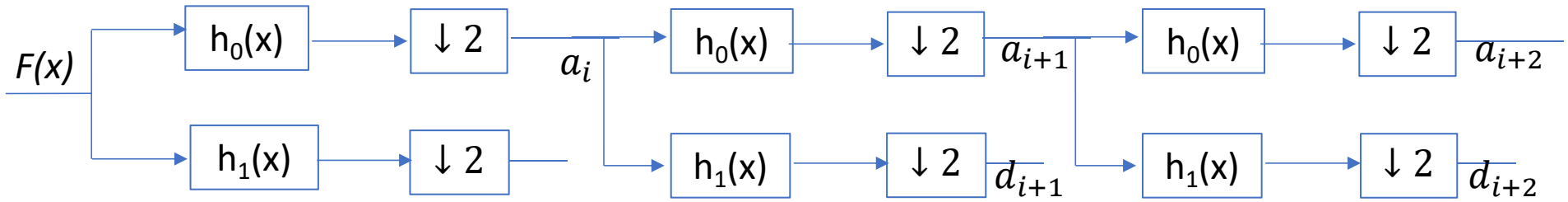


Faculty of Electrical Engineering, USTHB [akourgli@usthb.dz
http://perso.usthb.dz/~akourgli/

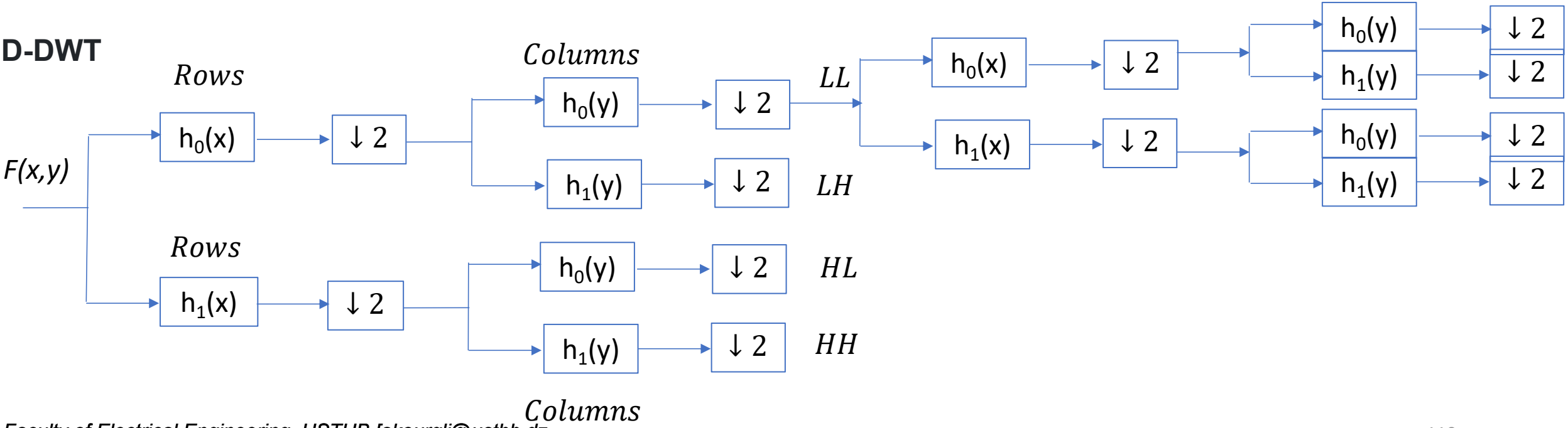
Transform Descriptors

DWT → 1 dimension

A signal is passed through two filters, high pass and low pass filters.



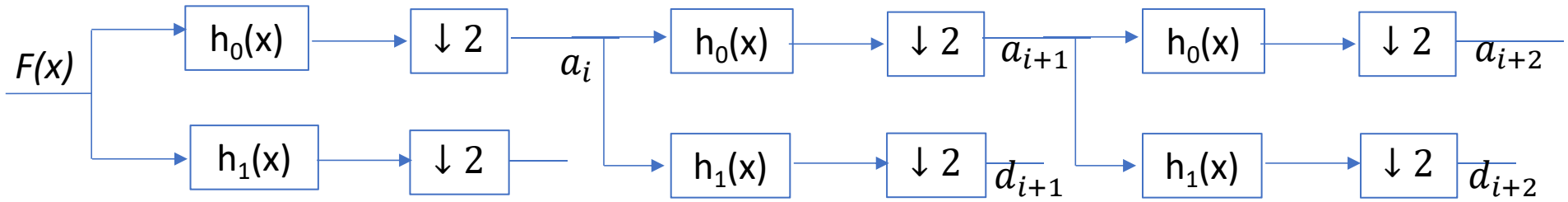
2D-DWT



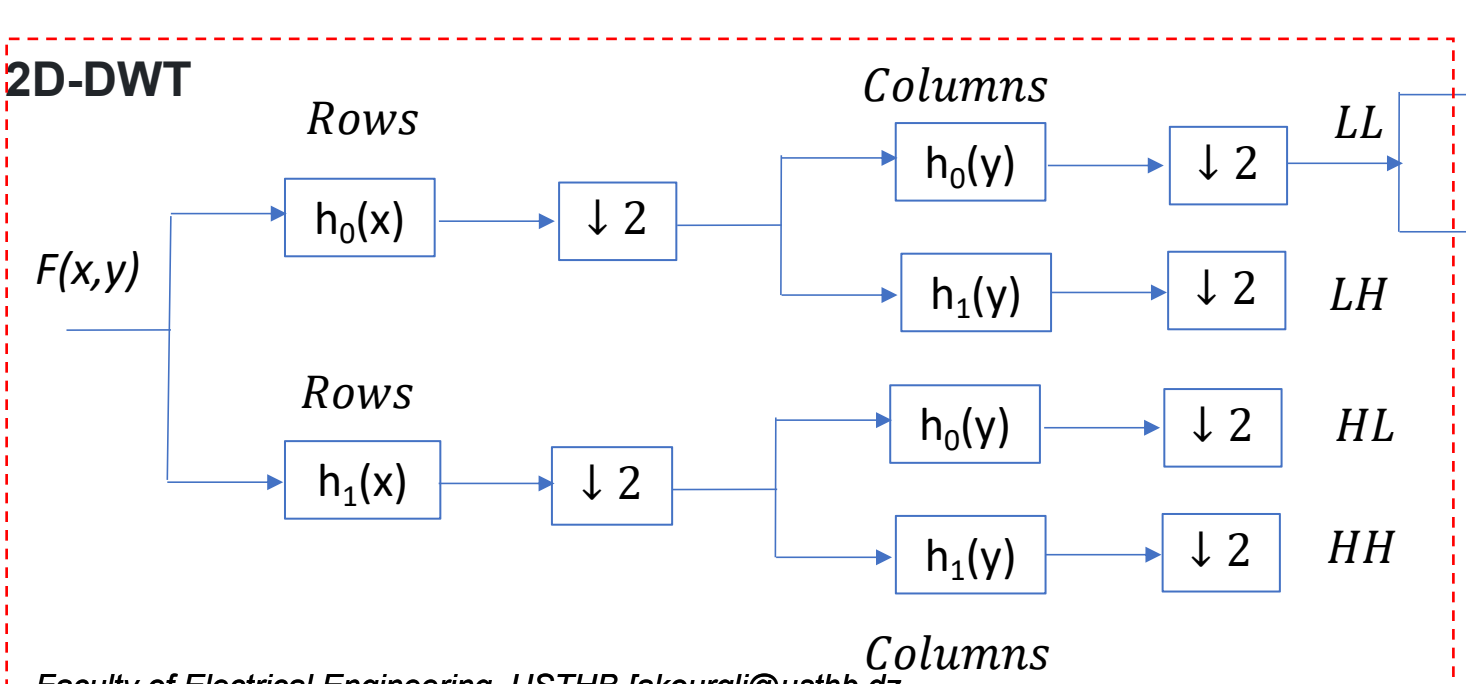
Transform Descriptors

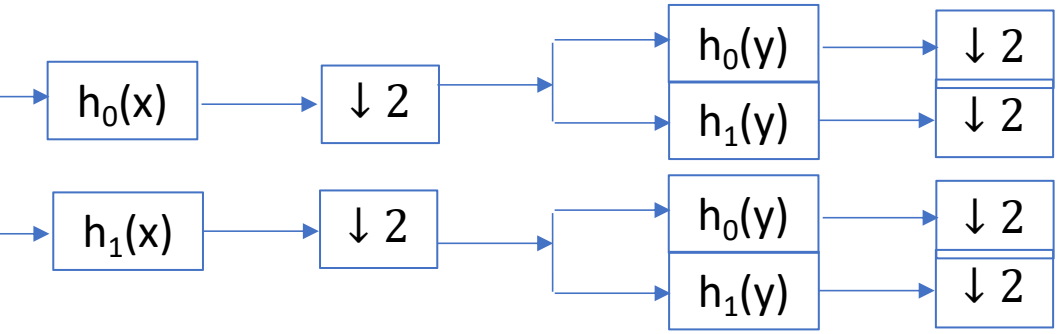
DWT → 1 dimension

A signal is passed through two filters, high pass and low pass filters.



2D-DWT

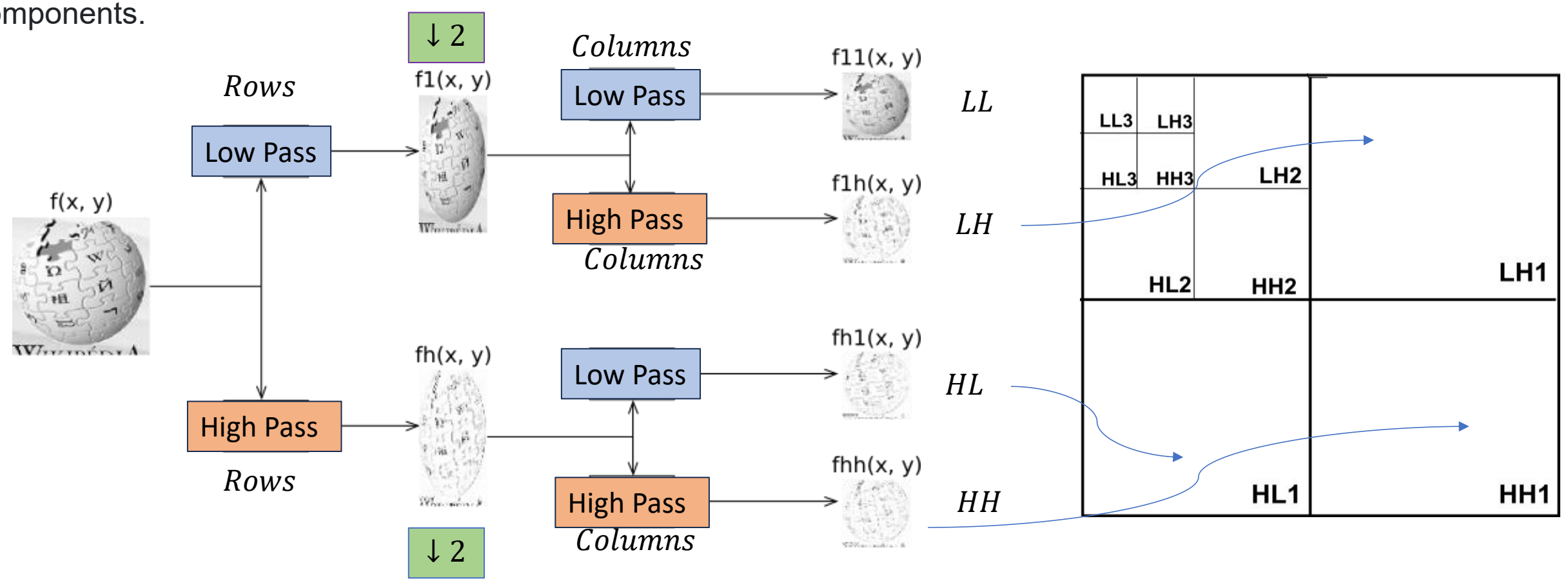




Transform Descriptors

2D-DWT

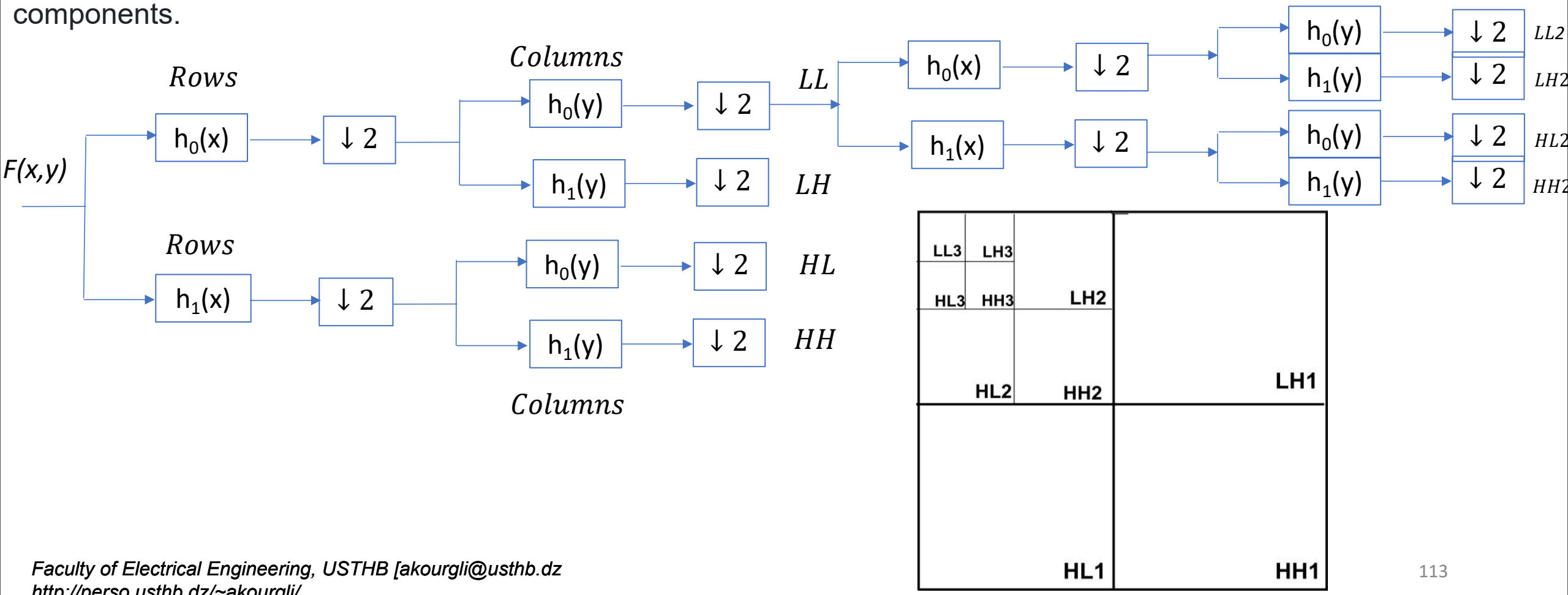
A signal is passed through two filters, high pass and low pass filters. The image is then decomposed into high frequency (details) and low frequency components (approximation). At every level, we get 4 sub-signals. The approximation shows an overall trend of pixel values and the details as the horizontal, vertical and diagonal components.



Transform Descriptors

2D-DWT

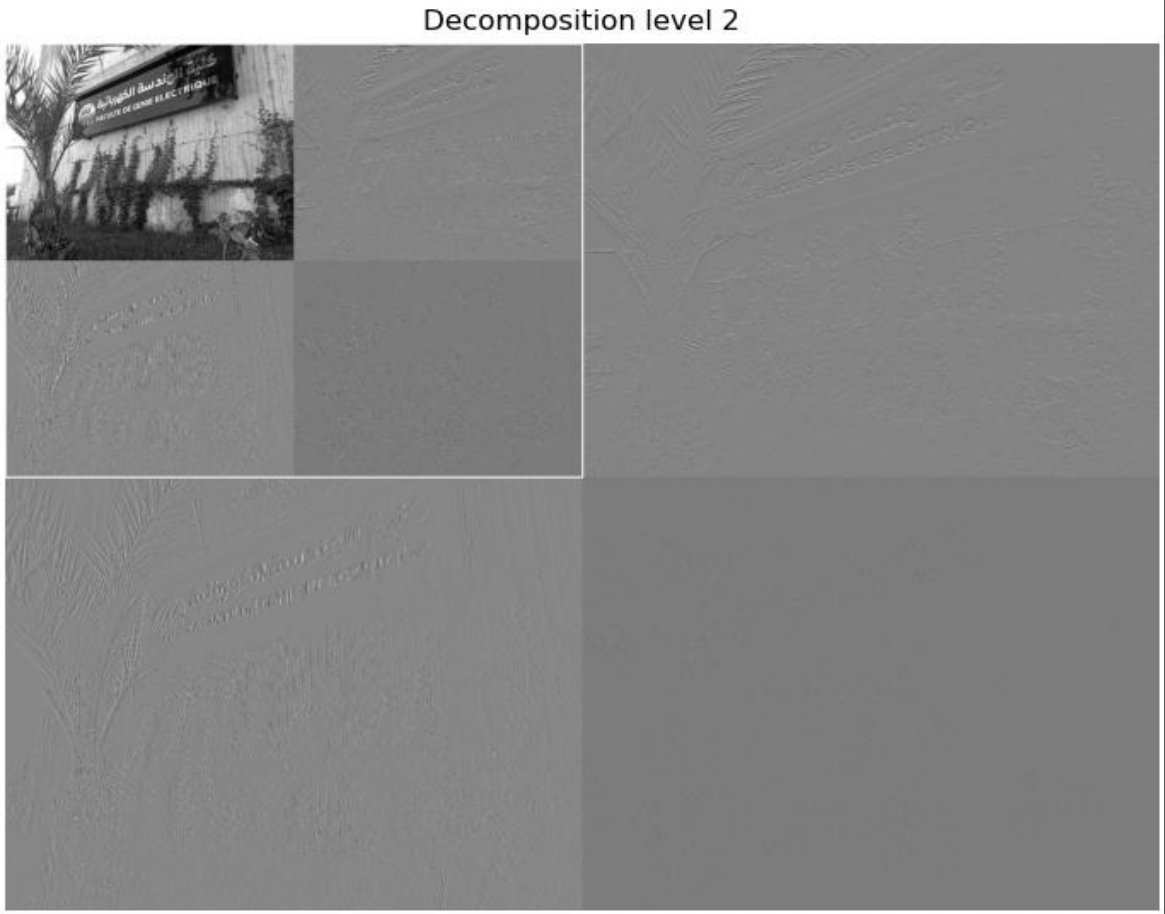
A signal is passed through two filters, high pass and low pass filters. The image is then decomposed into high frequency (details) and low frequency components (approximation). At every level, we get 4 sub-signals. The approximation shows an overall trend of pixel values and the details as the horizontal, vertical and diagonal components.



Transform Descriptors

2D-DWT

If these details are insignificant, they can be valued as zero without significant impact on the image, thereby achieving filtering and compression



Transform Descriptors

2D-DWT

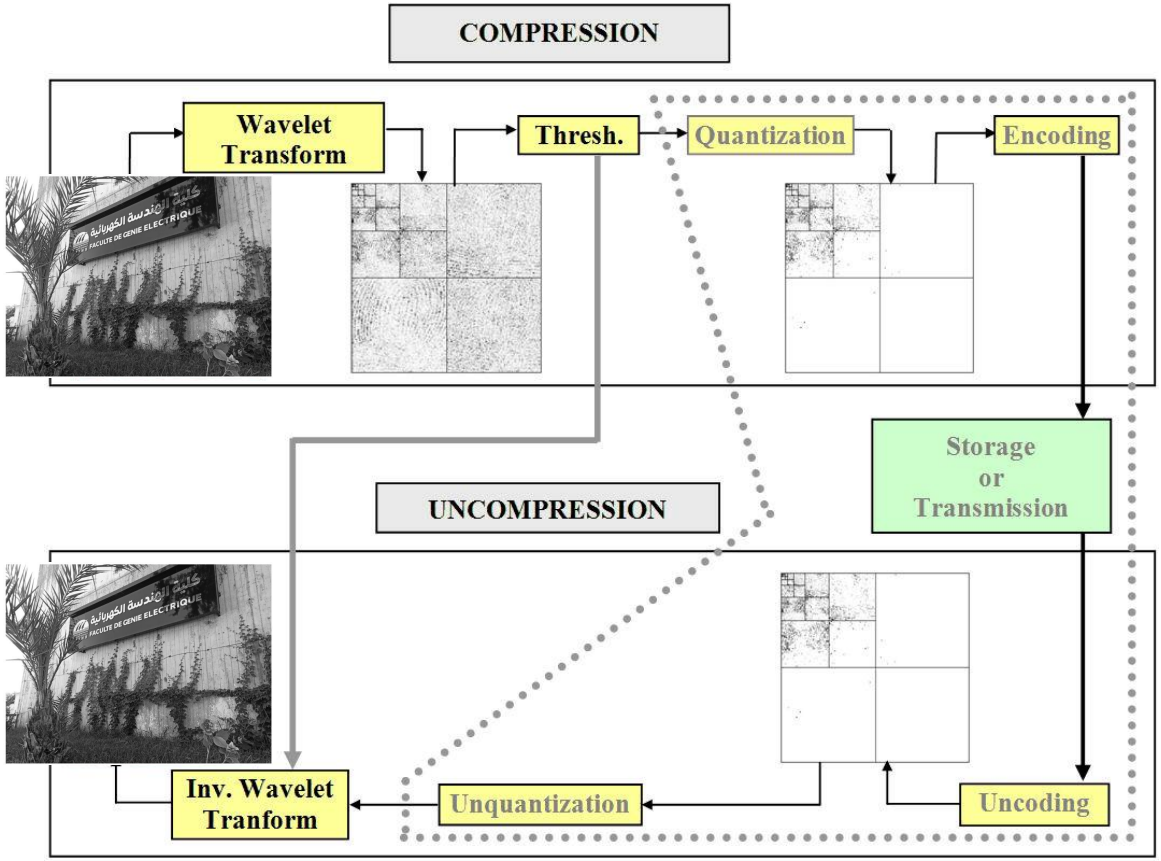
If these details are insignificant, they can be valued as zero without significant impact on the image, thereby achieving filtering and compression

Approximation level 2

Transform Descriptors

2D-DWT

If these details are insignificant, they can be valued as zero without significant impact on the image, thereby achieving filtering and compression



https://www.mathworks.com/help/wavelet/ug/wavelet-compression-for-images.html

Transform Descriptors

2D-DWT

If these details are insignificant, they can be valued as zero without significant impact on the image, thereby achieving filtering and compression

Decomposition level 2

Transform Descriptors

2D-DWT

If these details are insignificant, they can be valued as zero without significant impact on the image, thereby achieving filtering and compression

Compressed Image

Transform Descriptors

2D-DWT

If these details are insignificant, they can be valued as zero without significant impact on the image, thereby achieving filtering and compression

Denoised Image

Motion Descriptors

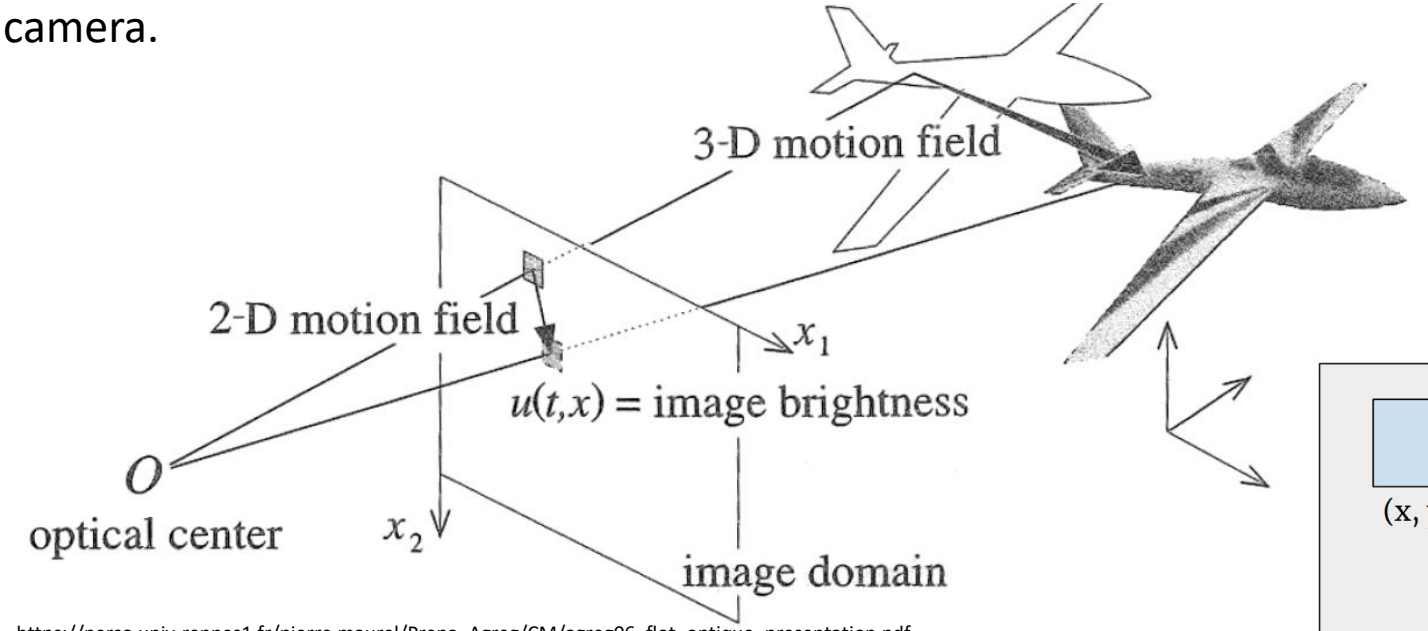
Aim : Processing real-time video input. Most techniques are based on addressing relationships of objects (x,y) within the same frame combined to time information *t*

Example :

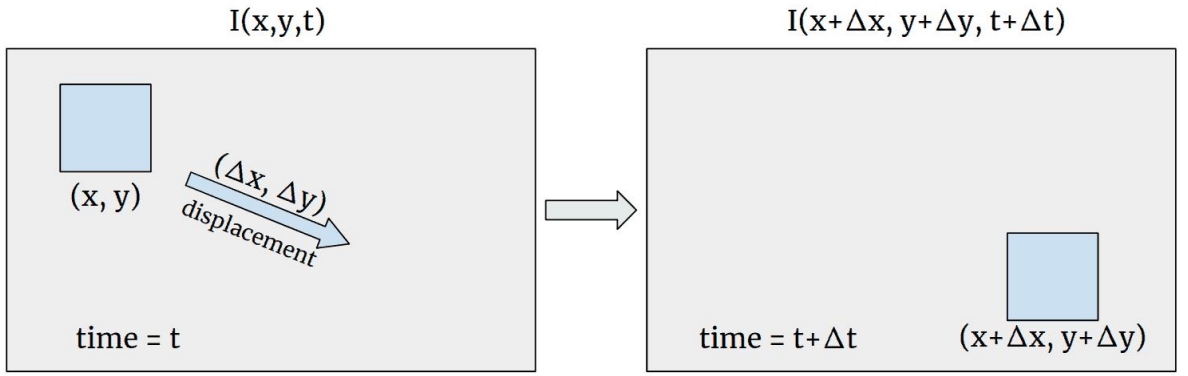
Track the motion of humans (pedestrians) or vehicles (cars, trucks, planes) across frames to **estimate** their current **velocity and predict** their **position** in the next frame or **recognize human actions.**

Optical flow :

Motion of objects between **consecutive frames** of sequence, caused by **the relative movement** between the object and the camera.



https://perso.univ-rennes1.fr/pierre.maurel/Prepa_Agreg/CM/agreg06_flot_optique_presentation.pdf



https://learnopencv.com/optical-flow-in-opencv/

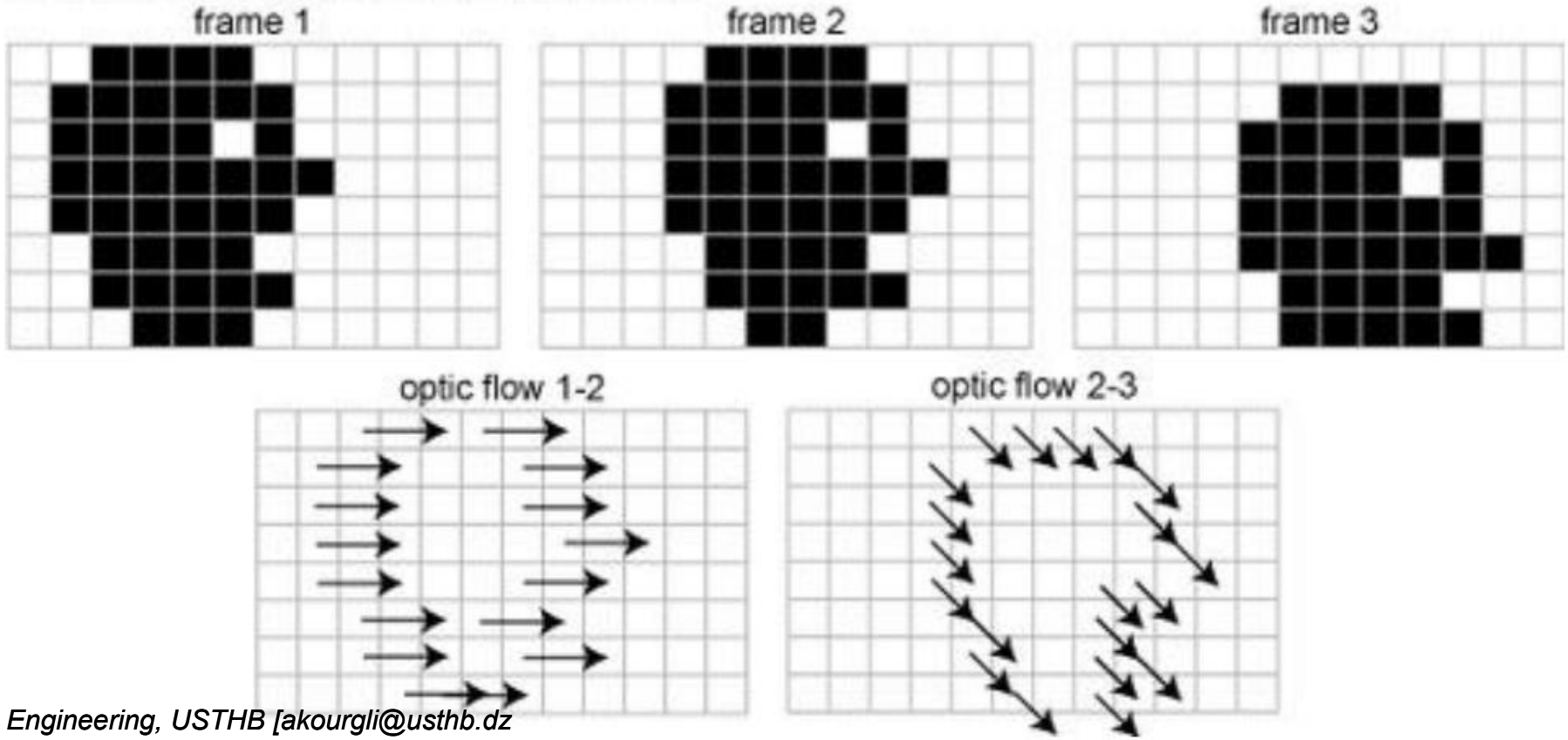
Motion Descriptors

Optical flow :

Motion of objects between **consecutive frames** of sequence, caused by **the relative movement** between the object and the camera.

We are looking for a field of **displacement vectors** representing the **Apparent movement** of each point.

$$I(x, y, t) - I_(x + \Delta x, y + \Delta y, t + \Delta t) = 0$$



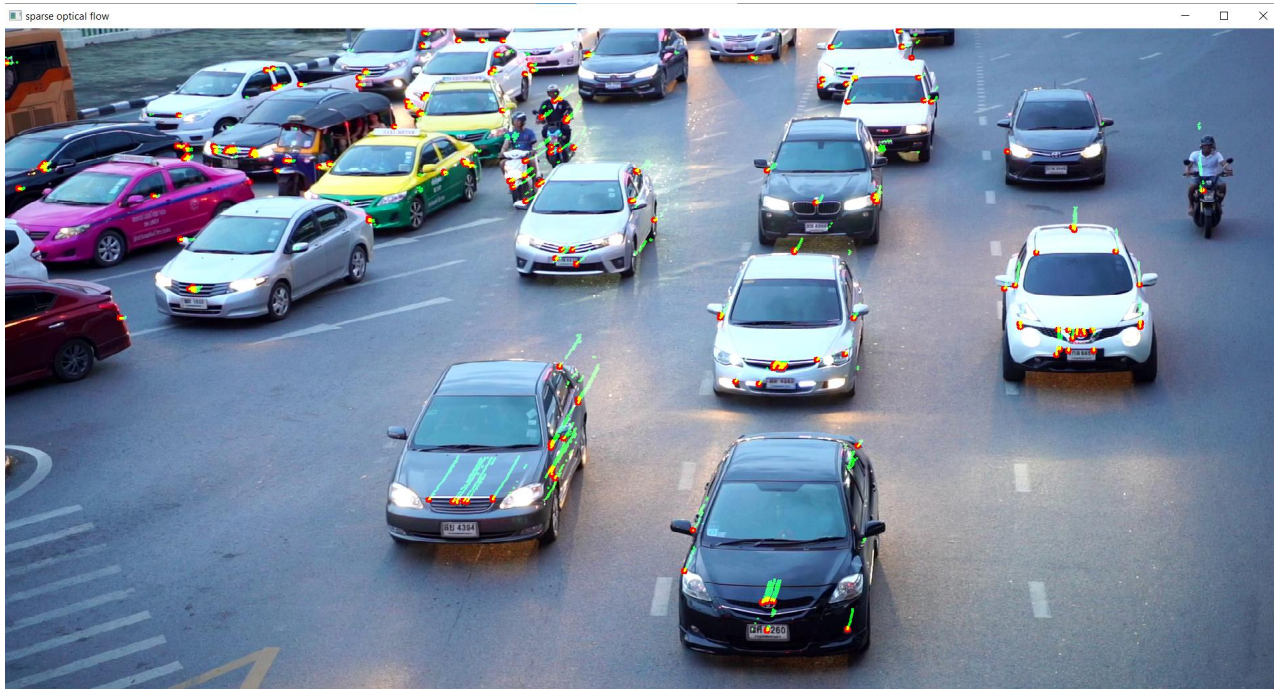
https://dspace.ummto.dz/server/api/core/bitstreams/4775228a-e188-4c78-8924-83bbc37746a6/content

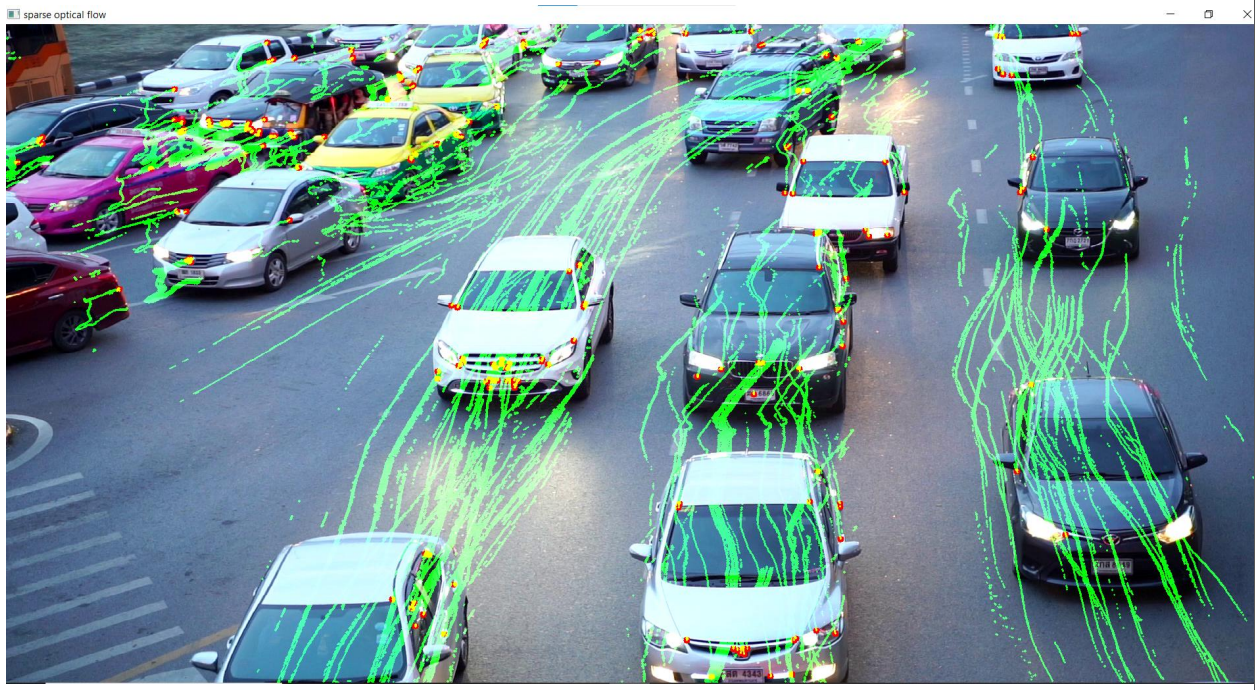
Motion Descriptors

Optical flow :

Motion of objects between **consecutive frames** of sequence, caused by **the relative movement** between the object and the camera.

We are looking for a field of **displacement vectors** representing the **Apparent movement** of each point.



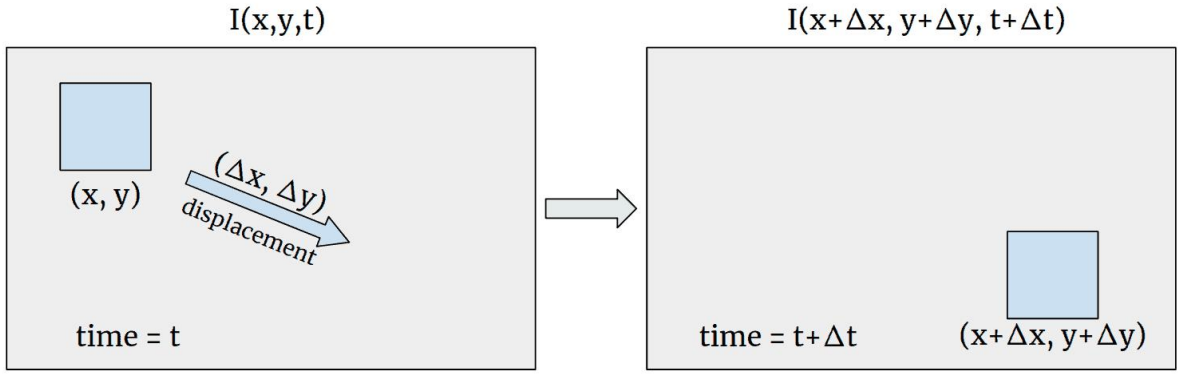


Motion Descriptors

Optical flow :

Motion of objects between **consecutive frames** of sequence, caused by **the relative movement** between the object and the camera.

The optical flow at time t and at point (x,y) is defined as the speed of the image point : $\left(\frac{dx}{dt}, \frac{dy}{dt}\right)$ → **Displacement vector**



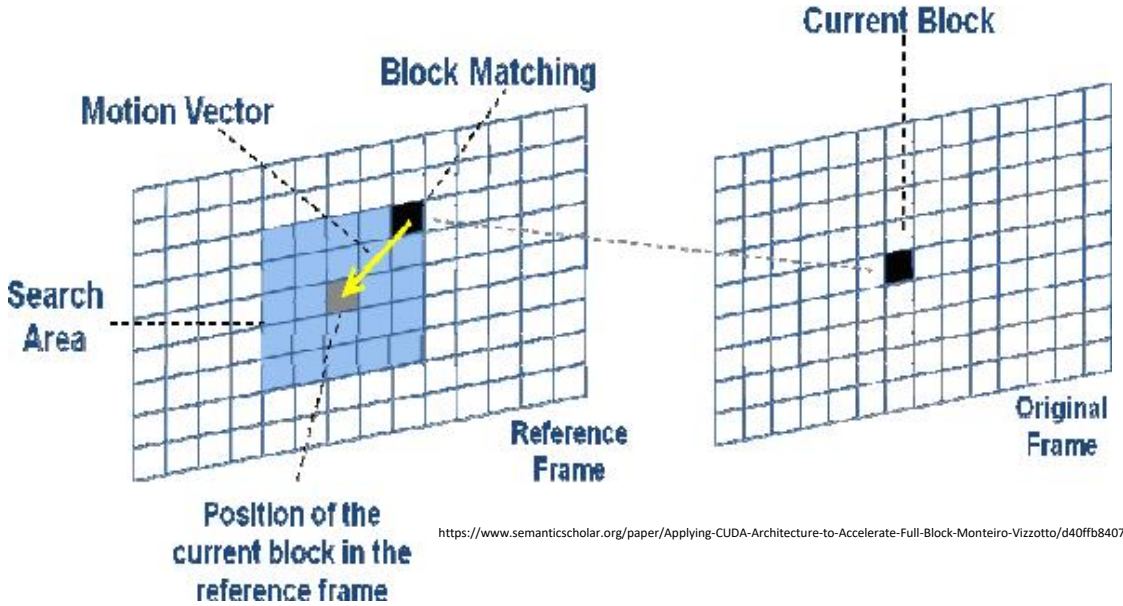
https://learnopencv.com/optical-flow-in-opencv/

We are looking for a field of **displacement vectors** representing the **Apparent movement** of each point.

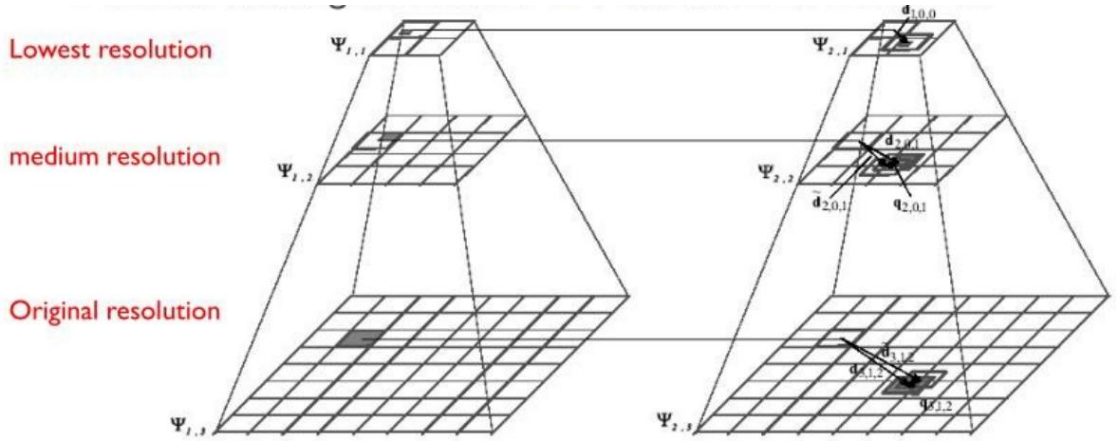
- *Block Matching* The speed is defined as the offset that gives the best match between regions image between frames.
- *Dense optical flow,* which gives the flow vectors of the entire frame (all pixels) - up to one flow vector per pixel.
- *Sparse optical flow* gives the flow vectors of some "interesting features" (few pixels depicting the edges or corners of an object) within the frame

Motion Descriptors

Block matching methods : The speed is defined as the offset that gives the best match (correlation) between regions image in different times. Finding the best match comes down to maximize a similarity measure, such as minimize a distance measure as the sum of squared differences (SDC).



https://www.semanticscholar.org/paper/Applying-CUDA-Architecture-to-Accelerate-Full-Block-Monteiro-Vizzotto/d40ffb8407ed088f7042e64764f367932ff33a2f



https://www.slideserve.com/file-download/1987727

Used for video compression (MPEG Norm) : Cutting into 16×16 or 32×32 blocks, Assuming of the same displacement for all pixels in the block., Coding: displacements + compressed errors/

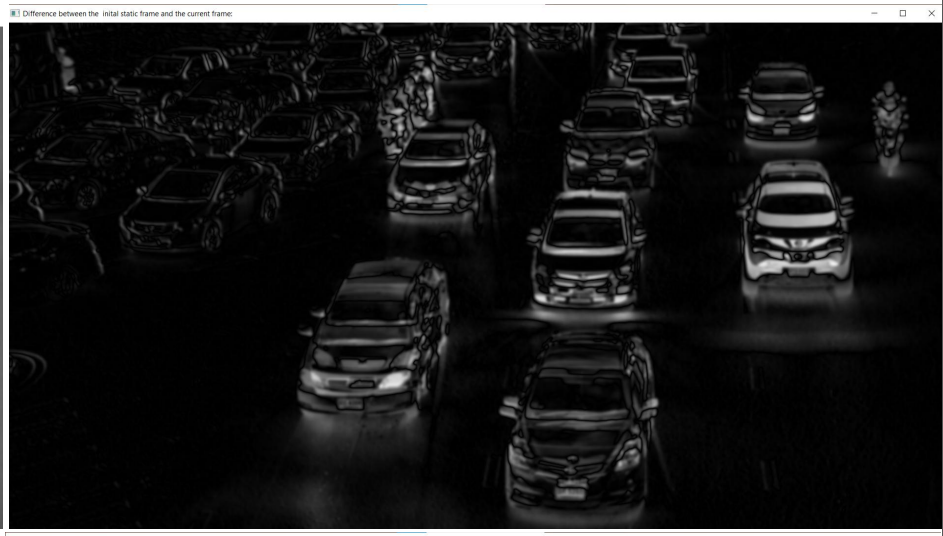
Other Applications : Segmentation, motion detection, prediction

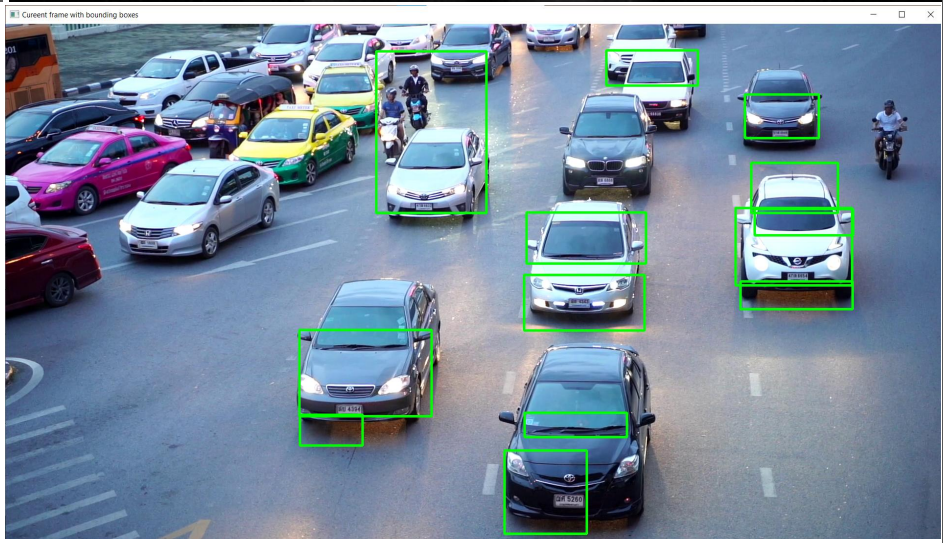
Motion Descriptors

Differential methods as Frame differentiating :

Three processing steps:

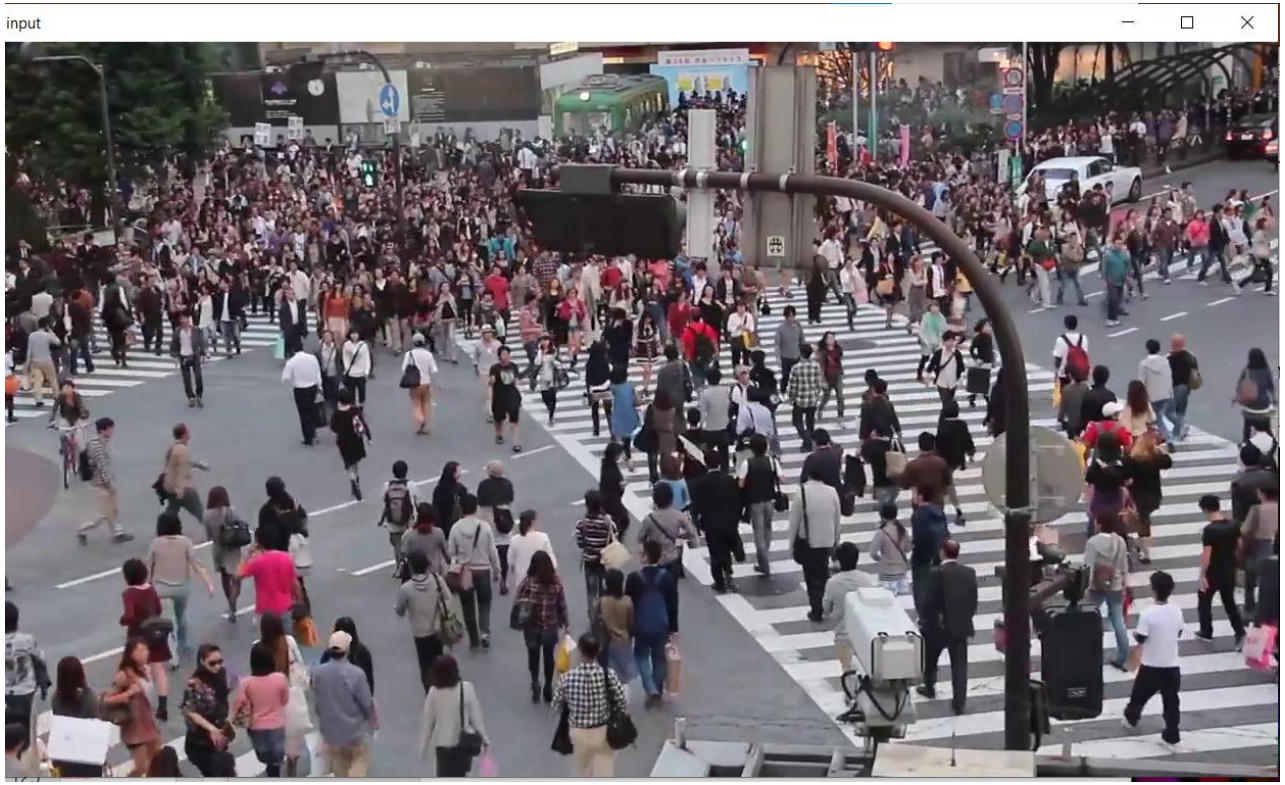
- ✓ Pre-filtering or smoothing with low-pas filters to obtain the signal structure of interest and improve the SNR.
- ✓ Extraction of basic measurements, such as space-time derivatives (to measure normal velocity components)
- ✓ Integrating these measurements to produce a 2-d optical flow adding assumptions about regularity.

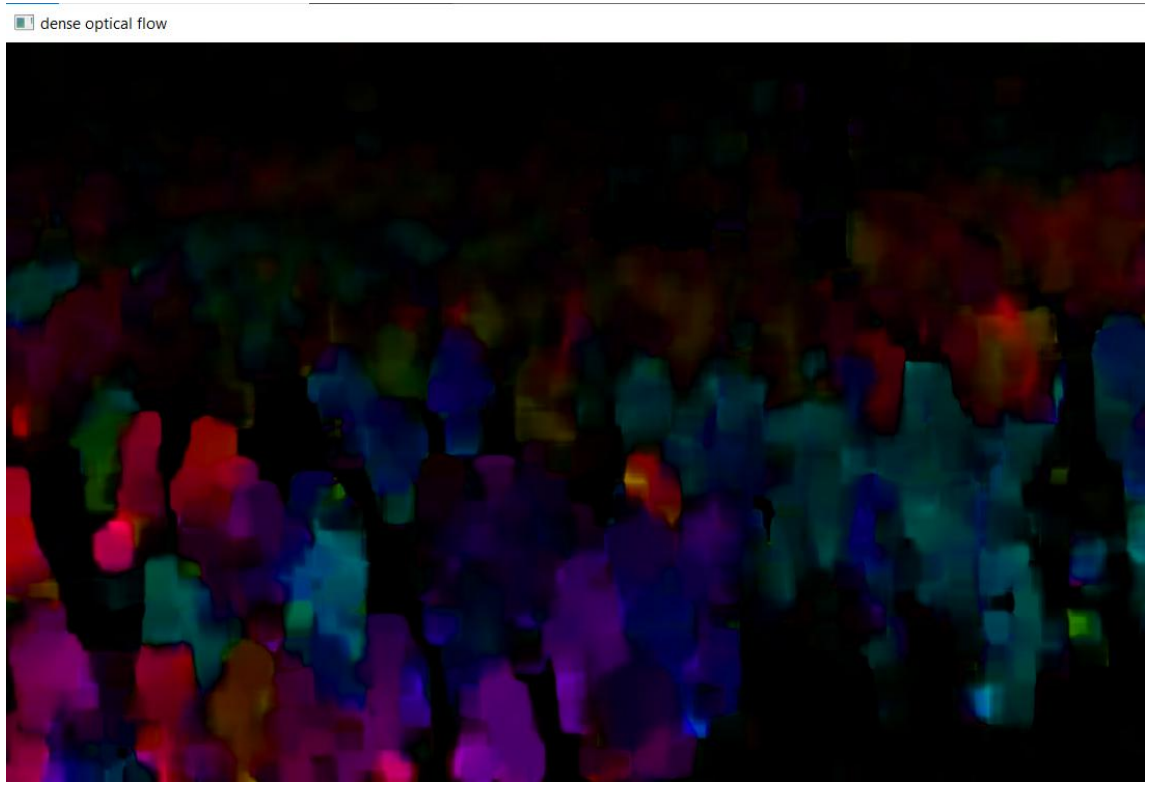




Motion Descriptors

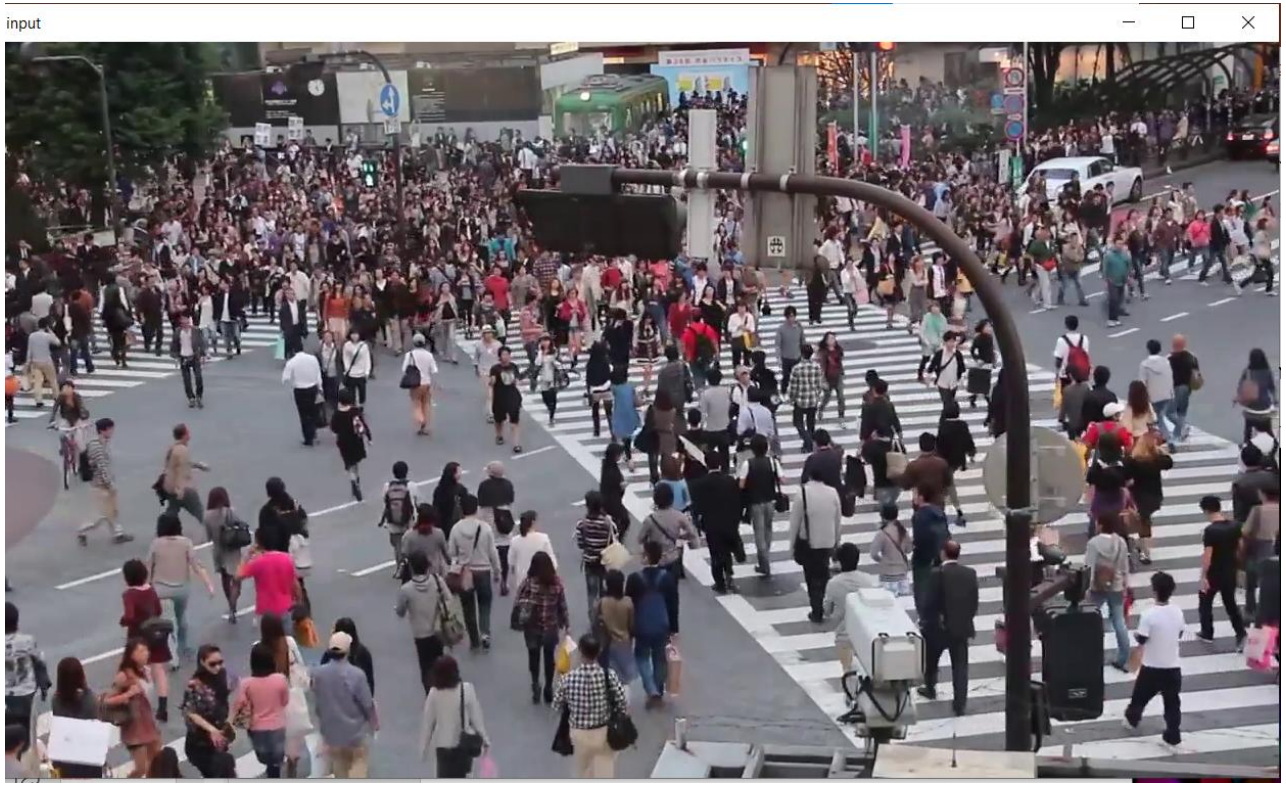
Farnback method : a **differential method** used for **dense** optical flow estimation. This method assumes that the flow is essentially constant in a local neighborhood of the pixel considered, and solves the optical flow equation for all pixels in this nei.ghborhood by the least squares method (Lucas–Kanade method).





Motion Descriptors

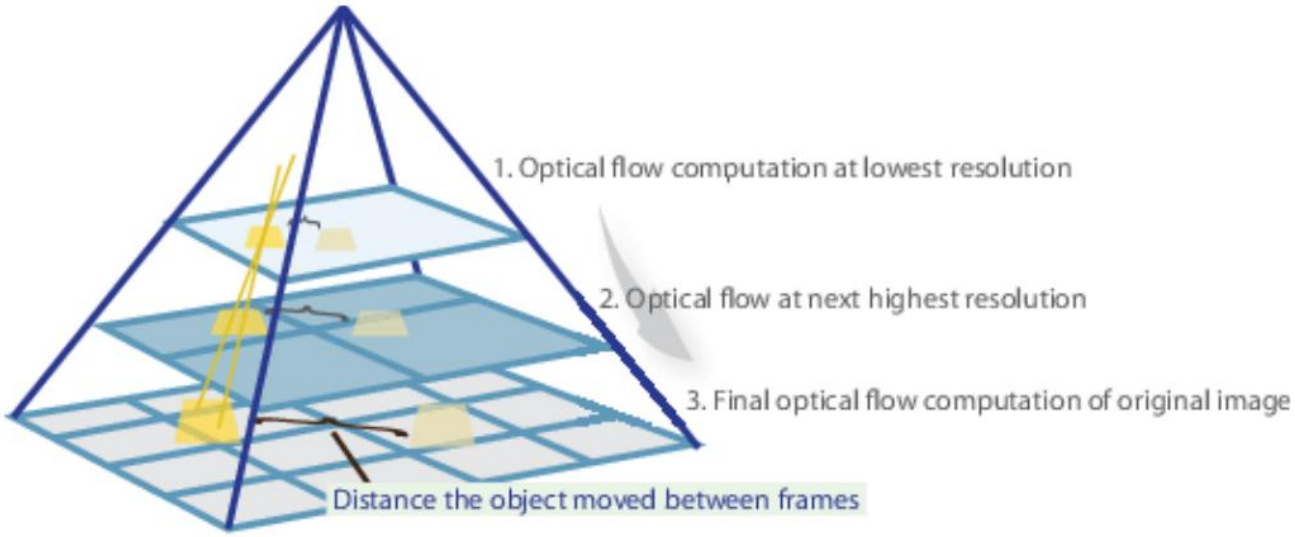
Pyramid Lucas Kanade : a **differential method** used for **sparse** optical flow estimation. It works only on **corner points** detected by Shi-Tomasi Algorithm (A modified version of Harris Detector). The extracted features are passed in the optical flow function from frame to frame to ensure that the **same points** are being **tracked**. There are various implementations of sparse optical flow: Horn–Schunck method, Buxton–Buxton method, etc.



Motion Descriptors

Pyramid Lucas Kanade : a **differential method** used for **sparse** optical flow estimation. It works only on **corner points** detected by Shi-Tomasi Algorithm (A modified version of Harris Detector). The extracted features are passed in the optical flow function from frame to frame to ensure that the **same points** are being **tracked**. There are various implementations of sparse optical flow: Horn–Schunck method, Buxton–Buxton method, etc.

A small 3x3 window (neighborhood) around the features detected by Shi-Tomasi and assume that all nine points have the same motion.



https://nanonets.com/blog/optical-flow/

- Dense Pyramid Lucas-Kanade
- Farneback
- PCAFlow
- SimpleFlow
- RLOF
- DeepFlow
- DualTVL1

Motion Descriptors

Optical Flow

https://pyimagesearch.com/2015/05/25/basic-motion-detection-and-tracking-with-python-and-opencv/
https://github.com/ic0n/Motion-Detection-Python/blob/master/motion_detection.py
https://www.kdnuggets.com/2022/08/perform-motion-detection-python.html
https://daweet.github.io/blog/2020/05/12/Python-Kinematics-Motion-Diagram-Tutorial
https://github.com/phausamann/rigid-body-motion
https://stackoverflow.com/questions/42582440/using-code-to-write-a-function-that-calculates-values-for-projectile-motion
https://nanonets.com/blog/optical-flow/
https://movingpandas.org/
https://learnopencv.com/optical-flow-in-opencv/
https://amroamroamro.github.io/mexopencv/opencv/opt_flow_demo.html
https://docs.opencv.org/4.x/dc/d6b/group__video__track.html
https://www.geeksforgeeks.org/opencv-the-gunnar-farneback-optical-flow/
https://helios2.mi.parisdescartes.fr/~lomn/Data/Rendus2022/Rapport_final_sequence_videopdf.pdf

Semestre : 2
Unité d'enseignement: UED 1.2
Matière 1 :Représentation des données Images et Vidéos
VHS : 22h30 (Cours : 1h30) Crédits : 1 Coefficient : 1

Chapitre 1. Introduction à la représentation d'images et de la vidéo (3 Semaines)

- ✓ Acquisition et formation d'une image, numérisation d'images.
- ✓ Espaces colorimétriques et transformations de couleur (RGB, HSV, YCrCb)
- ✓ Notions de résolution et quantification d'une image numérique (Taille, dpi, ppi, bpp, etc).
- ✓ Différents types d'images (Thermiques, Echos radar , satellitaires, images de capteurs sans fils, etc).
- ✓ Formats d'une image numérique (BMP, TIFF, JPG, GIF, et PNG).
- ✓ Notions de la vidéo numérique, formats vidéo .
- ✓ Conversion des formats vidéo et extraction de trames.

Semestre : 2
Unité d'enseignement: UED 1.2
Matière 1 :Représentation des données Images et Vidéos
VHS : 22h30 (Cours : 1h30) Crédits : 1 Coefficient : 1

Chapitre 4. Atelier de prise en main des outils de traitement (4 Semaines)

- ✓ Prise en main des outils de traitement d'images et de vidéo sous environnement C++ et OpenCV
- ✓ Lecture et écriture des fichiers images et vidéos
- ✓ Capture d'une séquence d'un fichier vidéo
- ✓ Représentation RVB , HSV, binaire, niveau de gris d'une image
- ✓ *Application des techniques de filtrage d'images*

Semestre : 2
Unité d'enseignement: UED 1.2
Matière 1 :Représentation des données Images et Vidéos
VHS : 22h30 (Cours : 1h30) Crédits : 1 Coefficient : 1

Chapitre 2. Traitement des images et des vidéos (4 semaines)

- ✓ Chaine de traitement de l'image et de la vidéo
- ✓ *Méthodes de prétraitement : Amélioration de la qualité (manipulation d'histogramme, restauration et réduction du bruit, détection de contours)*
- ✓ Notions de post-traitements

Chapitre 3. Représentation de l'image et vidéo pour l'analyse (4 semaines)

- ✓ Représentation des images fixes : couleur, texture et forme.
- ✓ Descripteurs : couleurs, forme et textures.
- ✓ Descripteurs basés transformée (DCT et ondelettes)
- ✓ Descripteurs de mouvements
- ✓ Applications à l'analyse de l'image et de la vidéo

Semestre : 2
Unité d'enseignement: UED 1.2
Matière 1 :Représentation des données Images et Vidéos
VHS : 22h30 (Cours : 1h30) Crédits : 1 Coefficient : 1

OpenCV est la bibliothèque OpenSource de référence en ce qui concerne la vision par ordinateur (ou computer vision en anglais). En 22 ans d'existence, elle a accumulé plus de 2500 algorithmes optimisés issus de la littérature scientifique tels que SIFT, les cascades de Haar, ou encore la transformée de Hough. Elle permet entre autre :

- L'ouverture, la modification, l'enregistrement et l'affichage de fichiers images et vidéos.
- L'extraction d'informations sur la répartition statistiques des pixel dans une image (moyenne, variance, histogramme)
- La segmentation d'image (Seuillage d'Otsu, K-moyennes,...).
- La localisation et l'extraction d'objets.
- L'extraction d'informations (Aire, périmètre, couleurs, ...) sur les objets contenus dans une images.
- La détection de contours (Filtre de Sobel, laplacien, Scharr, ...), de ligne ou de cercle (Transformée de Hough).
- Opérations d'image usuelles (redimensionnement, rotation, opérateurs morphologiques, filtrage, etc .....).
- Détection et suivi en temps réels d'objets (yeux, voiture, balle, ...) grâce à des algorithmes tels que les cascade de haar ou encore mean-shift.
- La mise en œuvre d'algorithmes usuels du machine learning tels que par le perceptron multicouches ou encore les arbres de décisions.

Annexe

Module PIL

```
from PIL import Image
A = Image.open("chemin_fichier")          ouverture d'un fichier image
A.load()                                  force le chargement de l'image en mémoire
A.mode                                    mode de l'image : 'L' en niveaux de
gris, 'RGB' en couleurs
A.format                                  format de l'image
A.size                                    taille de l'image sous la forme (Largeur,Hauteur)
A.save("chemin_fichier")                  sauvegarde d'une image dans un fichier
A = Image.new('RGB',(L,H))                création d'une image 'RGB' de dimensions
(Largeur,Hauteur)
r,g,b = A.split()                         récupère les composantes (r,g,b) de l'image
B = A.getpixel((x,y))                     lecture du pixel de coordonnées (x,y)
A.putpixel((x,y),(r,g,b))                 écriture de la valeur (r,g,b) dans le pixel (x,y
```

Annexe

Librairies à installer

pip install opencv-python

Liens

https://docs.opencv.org/4.x/d7/da8/tutorial_table_of_content_imgproc.html

https://docs.opencv.org/4.x/modules.html

https://dontrepeatyourself.org/post/how-to-read-and-write-videos-with-opencv

https://www.analyticsvidhya.com/blog/2021/04/top-python-libraries-for-image-processing-in-2021/

Annexe

RGB to HSV

1. R' = R/255

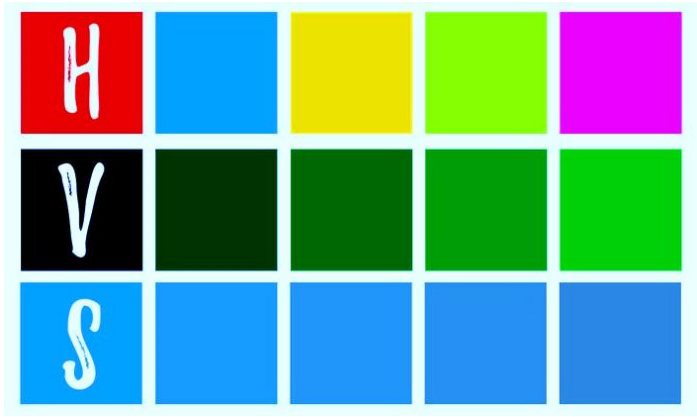
   G' = G/255

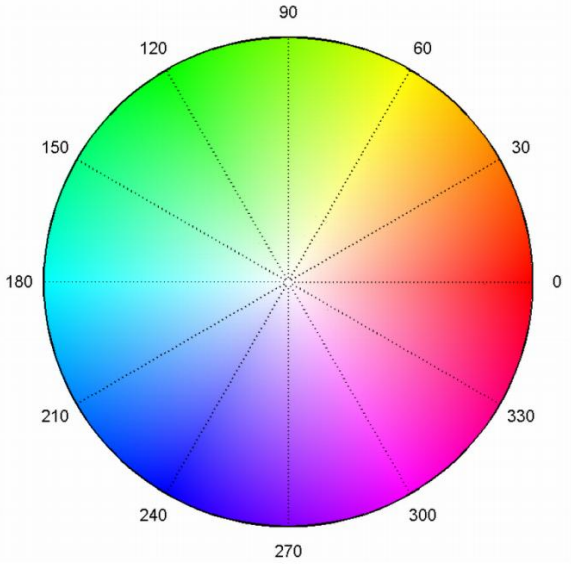
   B' = B/255*

2. Cmax = max(R', G', B')

   Cmin = min(R', G', B')

   Δ = Cmax - Cmin





3. $V = Cmax$

$$S = \begin{cases} \frac{\Delta}{Cmax} & si \quad Cmax \neq 0 \\ 0 & si \quad Cmax = 0 \end{cases}$$

$$H = \begin{cases} 60^0 \ \frac{G'-B'}{\Delta} + 360^0 (mod 360^0) & si \text{Cmax} = R' \\ 60^0 \ \frac{B'-R'}{\Delta} + 120^0 & si \text{Cmax} = G' \\ 60^0 \ \frac{R'-G'}{\Delta} + 240^0 & si \text{Cmax} = B' \\ 0 & si \text{Cmax} = Cmin \end{cases}$$

RGB to YUV

Y = 0.299 R + 0.587 G + 0.114 B U'= (B-Y)*0.565 V'= (R-Y)*0.713

Annexe

Caractéristique de la texture	Description	Formule
1. **L'énergie**(ou moment angulaire second)	L'énergie mesure l'homogénéité de l'image. Plus cette valeur est faible, moins l'image est uniforme et l'image contient beaucoup de variation de couleur. (i , j) : coordonnées dans la matrice. P(i , j) : valeurs normalisées de la matrice. Valeur dans l'intervalle [0,1].	$f1 = \sum_i \sum_j P(i,j)^2$
2. **Le contraste** ou inertie)	L'inertie mesure les variations locales des couleurs. Si ces variations sont importantes, alors le contraste sera élevé. Ce paramètre est fortement non corrélé à l'énergie. Valeur dans l'intervalle $[0,(Ng-1)^2]$.	$f2 = \sum_{i=1}^{Ng} \sum_{j=1}^{Ng} (i-j)^2 P(i,j)$
3. **Corrélation**	Où μ_x, μ_y, σ_x et σ_y sont respectivement les moyennes et les écarts type de p_x et p_y. Ce paramètre permet de déterminer si certaines colonnes de la matrice sont égales, c'est-à-dire s'il existe des dépendances linéaires dans l'image. En effet, plus les valeurs sont uniformément distribuées dans la matrice de cooccurrences et plus la corrélation est importante. Valeur dans l'intervalle [-1,1]. La corrélation n'est corrélée ni à l'énergie, ni à l'entropie.	$f3 = \frac{\sum_{i=1}^{Ng} \sum_{j=1}^{Ng} (ij) P(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y}$
4. **La variance**	La variance mesure la dispersion des valeurs autour de la moyenne. Plus ce paramètre est élevé et plus importants sont les écarts entre les valeurs et la moyenne. Elle représente l'hétérogénéité de la texture.	$f4 = \sum_i \sum_j (i-\mu)^2 P(i,j)$
5. **Moment différentiel inverse (Homogénéité)**	Ce paramètre mesure l'homogénéité de l'image, plus l'image contient de régions homogène, plus il est élevé. Valeur dans l'intervalle [0,1].	$f5 = \sum_i \sum_j \frac{P(i,j)}{1+(i-j)^2}$
6. **Moyenne des sommes**		$f6 = \sum_{i=1}^{2Ng} i.P_{x+y}(i)$
7. **Variance des sommes**		$f7 = \sum_{i=1}^{2Ng} (i-f8)^2.P_{x+y}(i)$
8. **Entropie des sommes**		$f8 = \sum_{i=1}^{2Ng} P_{x+y}(i).\log(P_{x+y}(i))$
9. **Entropie**	Ce paramètre mesure le désordre dans l'image.sa valeur sera élevé en cas de texture complètement aléatoire (sans structure apparente). Lorsque les valeurs de la matrice de cooccurrences sont presque toutes égales, l'entropie est élevée.	$f9 = -\sum_{i=1}^{Ng} \sum_{j=1}^{Ng} P(i,j).\log(P(i,j))$
10. **Variance des différences**		$f10 = \sum_{i=1}^{Ng} (\mu - \mu_{x-y})^2.P_{x-y}(i)$
11. **Entropie des différences**	Valeur dans l'intervalle [0,2.log(Ng-1)].	$f11 = \sum_{i=1}^{Ng} P_{x-y}(i) \log P_{x-y}(i)$
12. **Information sur la corrélation 1**	Soient : • Px, Py les probabilités selon x et y, et HX, HY leurs entropie respective. • $HXY = -\sum_{i=1}^{Ng} \sum_{j=1}^{Ng} P(i,j) \log(P(i,j))$ • $HXY1 = -\sum_{i=1}^{Ng} \sum_{j=1}^{Ng} P(i,j) \log(P_x(i)P_y(i))$ • $HXY2 = -\sum_{i=1}^{Ng} \sum_{j=1}^{Ng} P_x(i)P_y(j) \log(P_x(i)P_y(i))$	$f12 = \frac{(HXY - HXY1)}{Max(HX,HY)}$
13. **Information sur la corrélation 2**		$f13 = \sqrt{1 - e^{-2(HXY2-HXY)}}$
14. **Coefficient de corrélation maximal**	Soient la matrice : $Q(i;j) = \sum_k \frac{P(i,k)P(j,k)}{P_x(i)P_y(k)}$ $V = 2^{eme}$ Plus grande valeur propre de Q	$f14 = \sqrt{V}$

Basic Operations: Spatial Filtering

✓ Spatial filtering and 2D Convolution: mask concepts (average, Gaussian, binomial, etc.)

another image having different spatial and frequency properties :

- Linear: Replace the value of each pixel with a weighted average calculated with neighboring pixels. The mask contains the weighting coefficients of each pixel.

$$y(m,n) = \sum_i \sum_j h(i,j)x\,(m-i, n-j)$$

Example

h(-1,-1)	h(-1.0)	h(-1.0)
h(0,-1)	h(0,0)	h(0.1)
h(1,-1)	h(1,0)	h(1,1)

$$y(m,n) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} h(i,j)x(m-i, n-j)$$

x(0,0)	x(0,1)	x(0.2)	x(0.3)	x(0.4)
x(1,0)	x(1,1)	x(1,2)	x(1,3)	x(1,4)
x(2,0)	x(2,1)	x(2,2)	x(2,3)	x(2,4)
x(3,0)	x(3,1)	x(3,2)	x(3,3)	x(3,4)
x(4,0)	x(4,1)	x(4,2)	x(4,3)	x(4,4)

$$y(m,n) = \sum_i \sum_j g(i,j)x\,(m+i, n+j)$$

Applying the filter h(i,j) amounts to scanning the image with the mask g(i,j) = h(-i,-j)

Basic Operations: Spatial Filtering

✓ Spatial filtering and 2D Convolution: mask concepts (average, Gaussian, binomial, etc.)

another image having different spatial and frequency properties :

- Linear: Replace the value of each pixel with a weighted average calculated with neighboring pixels. The mask contains the weighting coefficients of each pixel.

$$y(m,n) = \sum_i \sum_j h(i,j)x\,(m-i, n-j)$$

Example

h(-1,-1)	h(-1.0)	h(-1.0)
h(0,-1)	h(0,0)	h(0.1)
h(1,-1)	h(1,0)	h(1,1)

$$y(m,n) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} h(i,j)x(m-i, n-j)$$

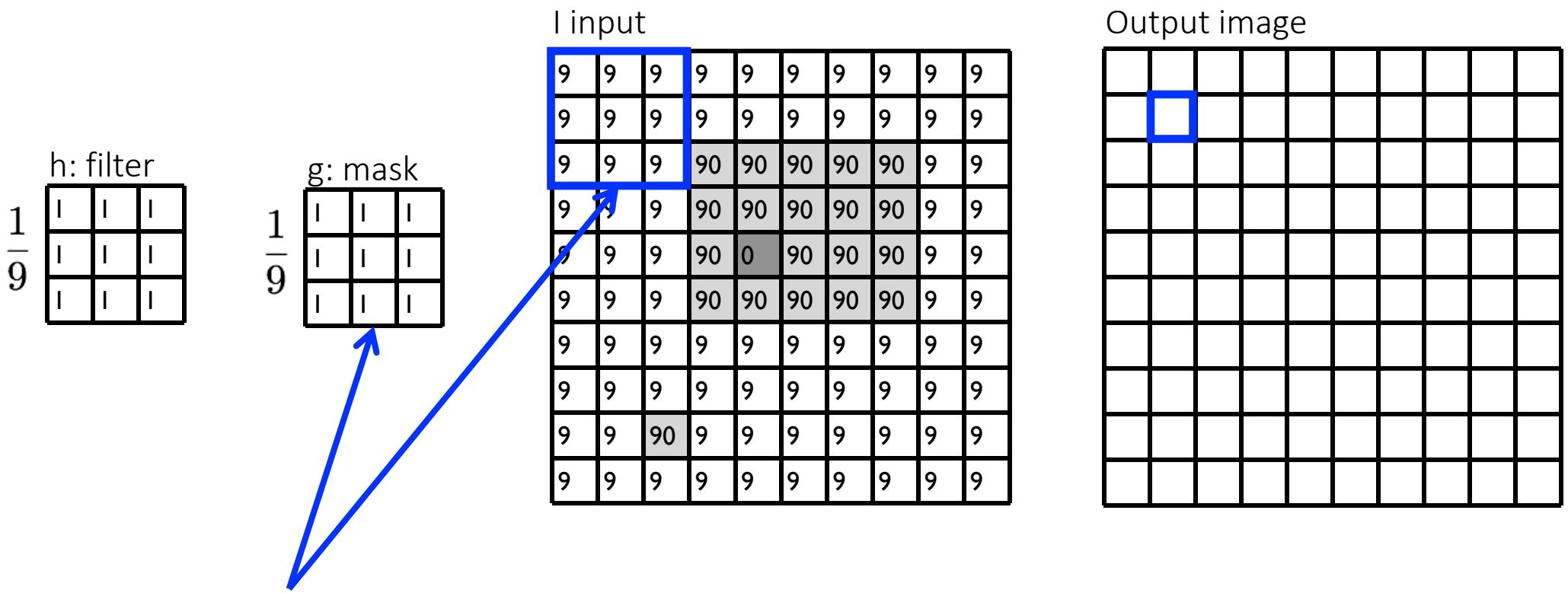
h(1,1)	h(1,0)	h(1,-1)
h(0.1)	h(0,0)	h(0,-1)
h(-1.0)	h(-1.0)	h(-1,-1)

x(0,0)	x(0,1)	x(0.2)	x(0.3)	x(0.4)
x(1,0)	x(1,1)	x(1,2)	x(1,3)	x(1,4)
x(2,0)	x(2,1)	x(2,2)	x(2,3)	x(2,4)
x(3,0)	x(3,1)	x(3,2)	x(3,3)	x(3,4)
x(4,0)	x(4,1)	x(4,2)	x(4,3)	x(4,4)

$$y(m,n) = \sum_i \sum_j g(i,j)x\,(m+i, n+j)$$

Applying the filter h(i,j) amounts to scanning the image with the mask g(i,j) = h(-i,-j)

Basic Operations: Spatial Filtering



Sweep the image with the mask $\quad g(i,j) = h(-i,-j)$

$$y(m,n) = \sum_i \sum_j h(i,j)x\,(m-i, n-j)$$

$$y(m,n) = \sum_i \sum_j g(i,j)x\,(m+i, n+j)$$

Basic Operations: Spatial Filtering

h: filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

g: mask

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9								

Basic Operations: Spatial Filtering

h: filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

g: mask

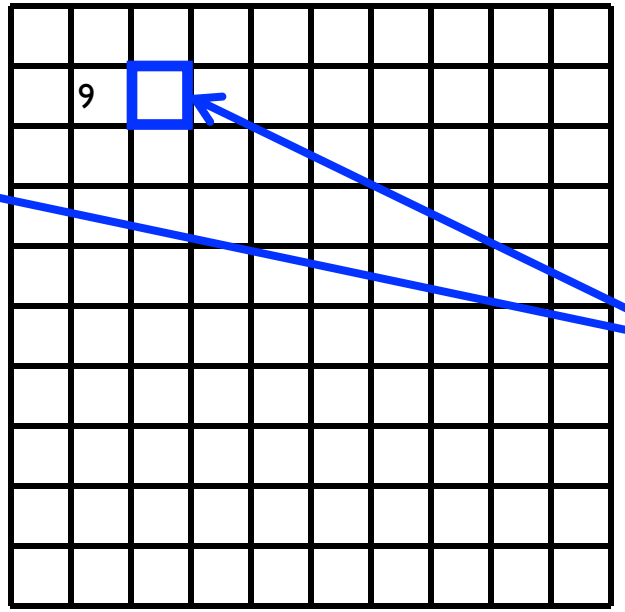
$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image



Basic Operations: Spatial Filtering

h: filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

g: mask

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	18							

Basic Operations: Spatial Filtering

h: filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

g: mask

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	18							

Basic Operations: Spatial Filtering

h: filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

g: mask

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	18	27						

Basic Operations: Spatial Filtering

h: filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

g: mask

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	18	27						

Basic Operations: Spatial Filtering

h: filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

g: mask

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	18	27	36					

Basic Operations: Spatial Filtering

h: filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

g: mask

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	18	27	36	36				

Basic Operations: Spatial Filtering

h: filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

g: mask

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	18	27	36	36	36	27		

Basic Operations: Spatial Filtering

h: filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

g: mask

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	18	27	36	36	36	27	18	

Basic Operations: Spatial Filtering

h: filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

g: mask

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	18	27	36	36	36	27	18	
	9								

Basic Operations: Spatial Filtering

h: filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

g: mask

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	18	27	36	36	36	27	18	
	9	27							

Basic Operations: Spatial Filtering

h: filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

g: mask

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	18	27	36	36	36	27	18	
	9	27	45						

Basic Operations: Spatial Filtering

h: filter

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

g: mask

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	18	27	36	36	36	27	18	
	9	27	45	63					

Basic Operations: Spatial Filtering

h: filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

g: mask

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	18	27	36	36	36	27	18	
	9	27	45	63	63	63	45	27	

Basic Operations: Spatial Filtering

h: filter

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

g: mask

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	18	27	36	36	36	27	18	
	9	27	45	63	63	63	45	27	
	9								

Basic Operations: Spatial Filtering

h: filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

g: mask

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	18	27	36	36	36	27	18	
	9	27	45	63	63	63	45	27	
	9	36	53	80	80	90	63	36	
	9	36	53	80	80	90	63	36	
	9	27	35	53	53	63	45	27	
	9	18	27	36	36	36	27	18	
	18	18	18	9	9	9	9	9	
	18								

Basic Operations: Spatial Filtering

h: filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

g: mask

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

Medium

I input

9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	90	0	90	90	90	9	9
9	9	9	90	90	90	90	90	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	90	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9

Output image

	9	18	27	36	36	36	27	18	
	9	27	45	63	63	63	45	27	
	9	36	53	80	80	90	63	36	
	9	36	53	80	80	90	63	36	
	9	27	35	53	53	63	45	27	
	9	18	27	36	36	36	27	18	
	18	18	18	9	9	9	9	9	
	18	18	18	9	9	9	9	9	

Texture and Shape descriptors

Textural Features

1. First Order Statistics/Statistical Features (FOS/SF)
2. Gray Level Co-occurence Matrix (GLCM/SGLDM)
3. Gray Level Difference Statistics (GLDS)
4. Neighborhood Gray Tone Difference Matrix (NGTDM)
5. Statistical Feature Matrix (SFM)
6. Law's Texture Energy Measures (LTE/TEM)
7. Fractal Dimension Texture Analysis (FDTA)
8. Gray Level Run Length Matrix (GLRLM)
9. Fourier Power Spectrum (FPS)
10. Shape Parameters
11. Gray Level Size Zone Matrix (GLSZM)
12. Higher Order Spectra (HOS)
13. Local Binary Pattern (LPB)

Morphological Features

1. Grayscale Morphological Analysis
2. Multilevel Binary Morphological Analysi

Histogram Based Features

1. Histogram
2. Multi-region histogram
3. Correlogram

Shape Features

1. Zernikes' Moments
2. Hu's Moments
3. Threshold Adjacency Matrix (TAS)
4. Histogram of Oriented Gradients (HOG)

Multi-scale Features

1. Fractal Dimension Texture Analysis (FDTA)
2. Amplitude Modulation – Frequency Modulation (AM-FM)
3. Discrete Wavelet Transform (DWT)
4. Stationary Wavelet Transform (SWT)
5. Wavelet Packets (WP)
6. Gabor Transform (GT)

https://pypi.org/project/pyfeats/
https://github.com/giakou4/pyfeats
https://pypi.org/project/pyfeats/0.0.11/