TS349: Pattern Recognition

Pattern Descriptors

Rémi Giraud

remi.giraud@enseirb-matmeca.fr

2023-2024

Bordeaux INP ENSEIRB MATMECA

ims

Objectives

- Understand the basic principles of shape and pattern recognition methods for image analysis.
- Know the associated vocabulary (shape, pattern, descriptor, feature extraction, local, dense, keypoints, invariance, etc...).
- Understand the basic principles of data classification and its applications in image processing.
- Know the specificities of the main unsupervised and supervised classification methods.
- Be able to implement, deploy and evaluate some of these approaches, validated on pattern recognition applications.

Organization

Pattern Descriptors:

Course	Shape descriptors & extraction	1h20
Practical n°1	Shape recognition (Hough Transform)	2h40
Course	Pattern descriptors & Dimension reduction	2h
Practical n°2	Texture classification (LBP, HOG)	2h40

Classification Methods:

Course	Unsupervised classification	2h
Practical n°1	Point cloud clustering	4h
Course	Supervised classification	2h40
Practical n°2	Digits classification	4h

Evaluation: Practicals (x0.25) + Final exam (x0.75) 1h20

Vocabulary

In English: "Pattern Recognition"   In French: "*Reconnaissance de Formes*"

A **shape** or **pattern** are simplified representations of a entity of the external world by an object manipulated by the computer (graph, vector, word, ...).

Examples: Characters, Digital print, Facial photography, Voice signal, ...

Pattern recognition (PR) consists of defining models allowing the automation of artificial perception tasks usually performed by the brain and the human sensory system.

Examples:
- Learn to recognize different patterns extracted from one or more observations
- Propose decisions based on the specificities of patterns

In this course, focus on PR in **image analysis** that includes the study of:

- **Shape**/*forme*, binary shape that can be defined by its contour
- **Pattern**/*motif* $=$ Region $+$ $\{$Properties$\}$ (color, orientation, repetition)

Applications et sectors

Production chains	- Posture control - Defect detection - Adaptation of treatments (fruit size, etc.)
Civil	- Text recognition (OCR) (postal sorting, radar control) - Autonomous driving - Home automation (voice recognition)
Life and earth sciences	- Recognition of fossil or organic species - Satellite imagery (crops, natural disasters, ...)
Security	- Detection of prohibited objects at boarding gates - Biometrics (fingerprints, vocal, retinal, facial, ...) - Target identification and tracking
Robotics	- Navigation
Medicine	- Anomaly detection - Surgical assistance

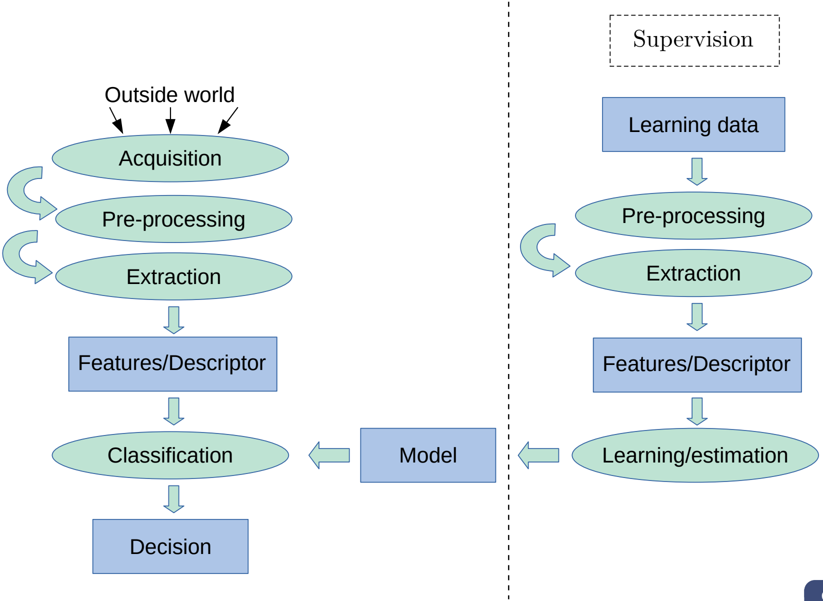
Machine perception

There are many examples of complex situations where the machine recognition process can fail:

- Contours based on human perception
- Information too degraded, noise, blur, illumination
- High intra-class variability
- High inter-class proximity, for example B and 13 have a similar shape

→ Learning process to identify the issues and understand the machine perception on these examples

PR conception cycle



Descriptor Properties

A “good” descriptor should provide: “similar” values for “similar” patterns
“different” values for “different” patterns

Properties:

- Rotation invariance: be able to identify the object, regardless of its orientation
- Translation invariance: even if the object is moved to a different location, it should be detected
- Scale invariance: should work even if the image is zoomed in or out
- Illumination invariance: should work even if there is change in brightness and contrast in the image
- Robustness to noise, deformations, occlusions
- Need for parameter tuning
- Time to compute
- Size
- ...

Classifier Properties

A “good” classifier should accurately predict the class corresponding to an input descriptor

Properties:

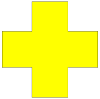
- Accuracy (on what evaluation metric?)
- Allowing errors
- Use/need of learning data
- Robustness to outliers (very different features compared to the dataset)
- Binary decision/class probabilities
- Fast to train/apply
- Need for parameter tuning

Shape vs Pattern Methods

In image analysis, different contexts referring to shape or pattern recognition

Shape

Binary shape



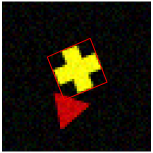
Contour descriptor

- **Freeman chain coding**
- **Fourier**
- **Signature**
- Tangent sequence
- Curvature scale space
- ...

Region descriptor

- **Structure**
- **Geometric**
- ...

Image containing shapes



Shape detection and extraction

- Radon transform
- **Hough transform**
- ...

Pattern

Local (keypoints)

Keypoint detectors

- **Harris corners**
- Shi-Tomasi corners
- FAST corners
- **SIFT keypoints**
- SURF keypoints
- ...

Local descriptors

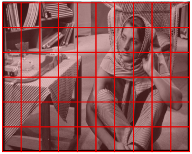
- **SIFT**
- SURF
- BRIEF
- ORB
- ...

Global (all pixels)

Global descriptors

- **Co-occurrence matrix**
- **Fourier transform**
- **Histogram of dense features (intensity, LBP, ...)**
- ...

Block-wise (all pixels)

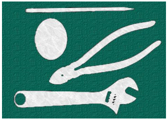


Block-wise descriptors

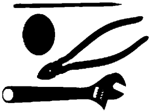
- **Block-wise histograms of dense features**
- **Histogram of Oriented Gradients (HOG)**
- Gradient Field HOG
- ...

Shape Recognition

- **Detection:**
  Localization and segmentation of the shape



Image

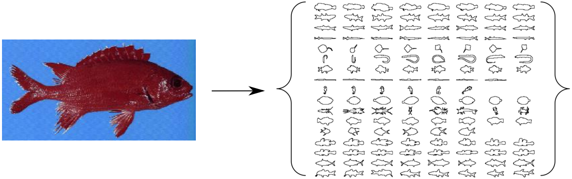


Binarization

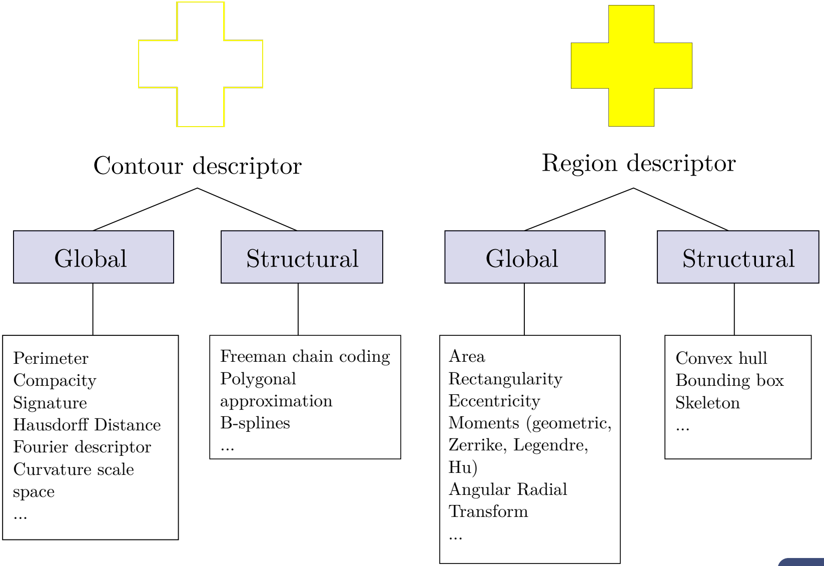
Connected components

Extraction (bounding box)

- **Description:**
  Describe the shape with an appropriate set of features (descriptor)
- **Identification/Classification/Recognition:**
  Find the class of the shape according to the problem (binary decision, class probabilities)

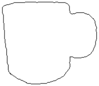


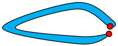
Taxonomy


Contour descriptor
Global
Structural
Perimeter
Compacity
Signature
Hausdorff Distance
Fourier descriptor
Curvature scale space
...
Freeman chain coding
Polygonal approximation
B-splines
...
Region descriptor
Global
Structural
Area
Rectangularity
Eccentricity
Moments (geometric, Zerrike, Legendre, Hu)
Angular Radial Transform
...
Convex hull
Bounding box
Skeleton
...

Contour vs Region

- Contour descriptor: Sequence of points (1D)
  - Cartesian, complex or polar representation. Freeman chain coding, polygons, etc...
  - Ambiguity on proximity of contour points



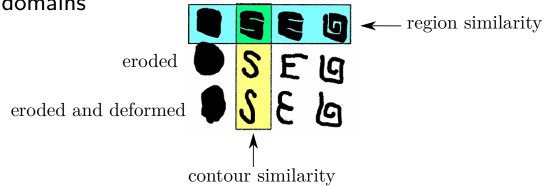


- Region descriptor: Pixel matrix (2D)
  - Geometric, moments, hull, skeletization, ...
  - Not equivalent to the contour-based approach

contours externes identiques

régions différentes

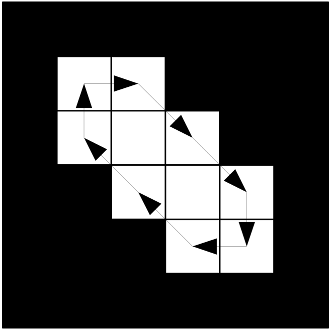
→ Blur domains



Freeman Chain Coding*

Definition: Coding of pixel movements along the contour of a shape

Motivation: Shapes can be entirely described by their contours. Such coding may serve to describe binary images.

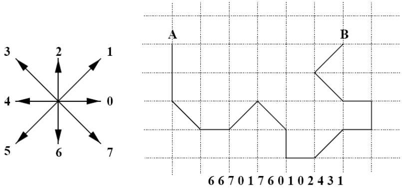


*Freeman et al., Computer Processing of Line-Drawing Images, ACM Computing Surveys, 1974

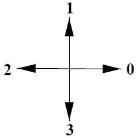
Freeman Chain Coding

- **Coding:** code = starting point + {movements}

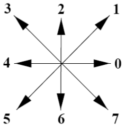
  Start from random point A, continue until coming back to A



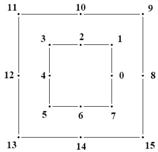
- **Resolution:** Limited number of local direction bits



4 directions (2 bits)



8 directions (3 bits)

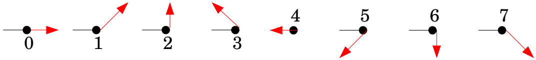


8 directions
16 movements (4 bits)

Freeman Chain Coding

- **Relative coding:** Coding of the change of direction instead of the direction

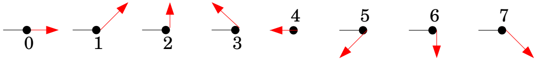
  code = starting point + first movement + {direction changes}



Freeman Chain Coding

- **Relative coding:** Coding of the change of direction instead of the direction

  code = starting point + first movement + {direction changes}

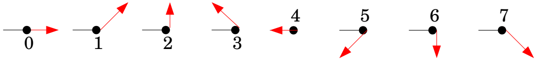


→ To reduce the entropy in the chain code (may increase the number of 0)

Freeman Chain Coding

- **Relative coding:** Coding of the change of direction instead of the direction

  code = starting point + first movement + {direction changes}



  → To reduce the entropy in the chain code (may increase the number of 0)

- **Properties**

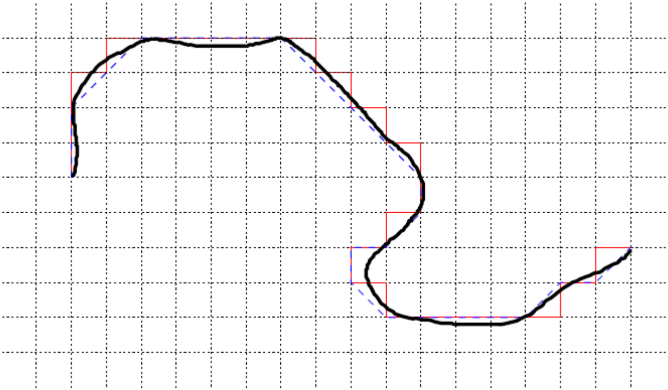
  Translation invariant
  "Rotation" (multiple of 360/P degrees) by addition (modulo P)
  "Dilation" by repetition
  Inversion (central symmetry) by complement

- **Comparison of chain codes**
  By measure of similarity of character strings
  Editing cost or distances (substitution, destruction, insertion)
  Wagner and Fisher algorithm, ...

- **Generalized version** for point sequences

Freeman Chain Coding

Example



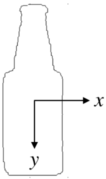
4 directions	1 1 1 0 1 0 0 0 0 0 0 3 0 3 0 3 0 3 3 2 3 2 3 0 3 0 0 0 0 0 1 0 1 0	34x2=68bits, entropy=1.66
8 directions	2 2 1 1 0 0 0 0 7 7 7 7 6 5 4 6 7 0 0 0 0 1 0 1	24x3=72bits, entropy=2.41
8 directions (relative)	2 0 7 0 7 0 0 0 7 0 0 0 7 7 7 2 1 1 0 0 0 1 7 1	24x3=72bits, entropy=1.76
16 movements	10 9 8 8 15 15 6 5 4 6 7 8 8 10 1	16x4=64bits, entropy=3.13

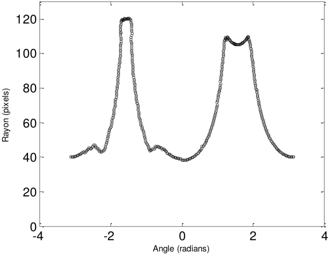
Signature

Definition: 1D vector representing a contour in polar space $(\boldsymbol{\rho}, \boldsymbol{\theta})$

Method:

Place the center of the system on the gravity center $(\bar{x}, \bar{y})$
For each angle $\theta \in [-\pi, \pi]$, compute the distance ρ to the closest point



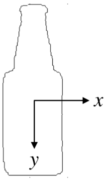


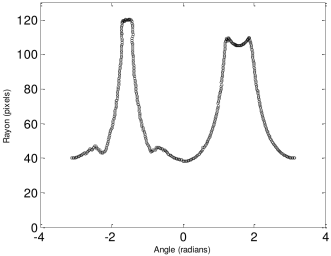
Signature

Definition: 1D vector representing a contour in polar space $(\boldsymbol{\rho}, \boldsymbol{\theta})$

Method:

Place the center of the system on the gravity center $(\bar{x}, \bar{y})$
For each angle $\theta \in [-\pi, \pi]$, compute the distance ρ to the closest point



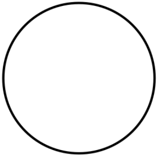


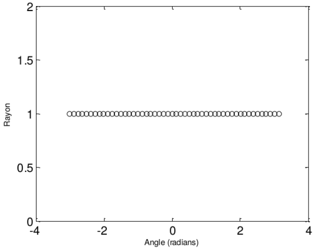
Properties:

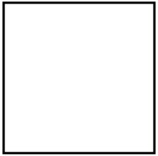
Invariance to translation
Rotation by translation of angles
Only adapted to star-convex shapes

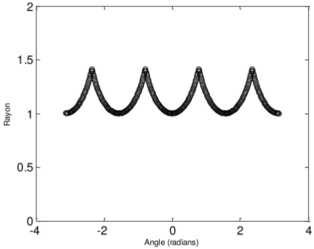
Signature

Examples:









Fourier Descriptors

Definition: The Fourier Transform of the shape's contour points

The contour is seen as a periodic sequence of N points $\{x_n, y_n\}$

- Complex form:

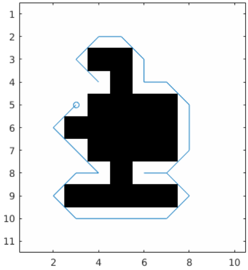
$$c_n = (x_n - \bar{x}) + j(y_n - \bar{y})$$

- Fourier Transform:

$$C_k = \frac{1}{N}\sum_{n=0}^{N-1} c_n \exp^{-2j\pi k \frac{n}{N}} \quad k \in [0, ..., N-1]$$

- Fourier Descriptors:

$$\text{FD} = \left\{ \frac{|C_2|}{|C_1|}, ..., \frac{|C_{N/2}|}{|C_1|} \right\}$$



Properties:

Relation between the number of coefficients and the details of approximation
The coefficients are ordered by their contributions (low→ high frequency)
Invariance to translation, rotation, and scaling

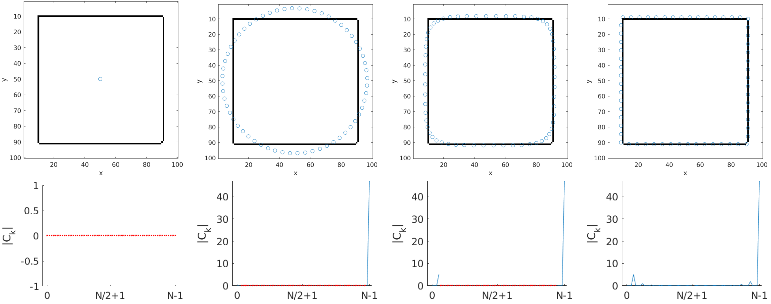
Fourier Descriptors

Reconstruction: Shape approximation by limited number K of FT coefficients

Inverse FT: $c_n = \sum_{n=0}^{N-1} C_k \exp^{2j\pi n \frac{k}{N}}$ with $n \in [0, ..., N-1]$

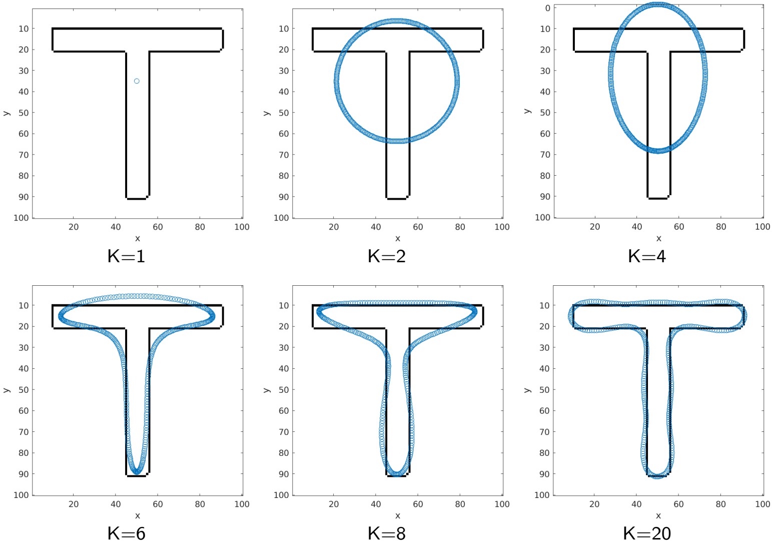
Low frequency components C_k with $k \approx 0$ and $k \approx N-1$ mainly contribute

Example with $N = 60$:



$K = 1$ $K = 4$ $K = 8$ $K = 60$

Example with $N = 560$:



Contour descriptors

Different methods with their advantages and limitations

- Freeman chain code
- Signature
- Fourier Descriptors
- See also Curve-based approaches:

  Tangent Sequence

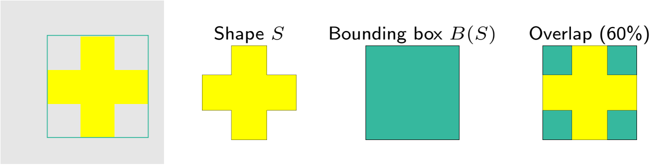
  Curvature Scale Space

→ Region descriptors

Structure

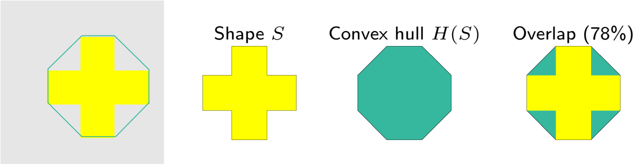
For a discrete 2D shape S

- **Bounding box:** The smallest rectangle that contains the shape



- **Convex Hull:** The smallest convex polygon that contains the shape

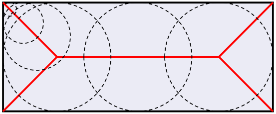
  Convex property: each line segment between two points of the polygon remains inside the polygon

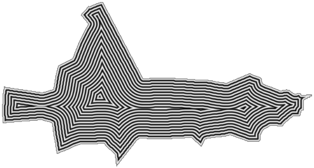


Structure

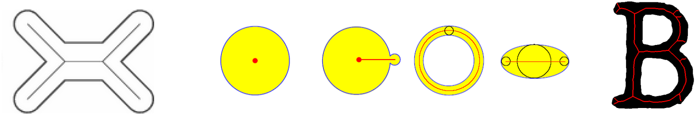
- **Skeleton:** Center of the tangential disks (inscribed) with maximal diameter

Can also be seen as "fire stopping line"





Examples:



→ The contour has an important impact on the skeleton

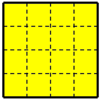
→ Used in physics and mechanics, medical image analysis (bronchus, eye retina), computer graphics (animation), path finding, etc.

Geometric

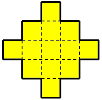
- **Area and perimeter:**

  Area $A(S)$ = Number of pixels in S

  Perimeter $P(S)$ = Number of contour pixels (outside) S



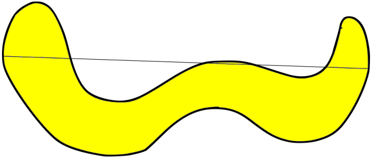
$A(S) = 16$
$P(S) = 16$



$A(S) = 13$
$P(S) = 20$

  $\rightarrow$ Very sensitive to sampling for small objects

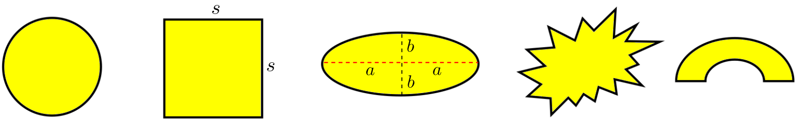
- **Diameter:** Largest distance between two points of S



Geometric

- **Compacity/Circularity:**

  Isoperimetric quotient $C(S) = \dfrac{4\pi A(S)}{P(S)^2}$

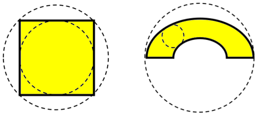


$C(S) = 1$ $\quad C(S) = \frac{4\pi s^2}{(4s)^2} = \frac{\pi}{4}$ $\quad C(S) \approx \frac{1}{2}\left(\frac{a}{b} + \frac{b}{a}\right)$ $\quad C(S) << 1$

$\rightarrow$ Invariant to rotation and scale (only in continuous domain)

- **Lengthening:**

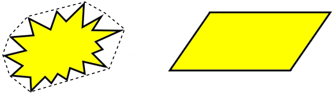
$$L(S) = \frac{\text{radius of the largest inscribed circle}}{\text{radius of the smallest circumscribed circle}}$$



Geometric

- **Concavity:** Ratio between the perimeters of the shape and its convex hull

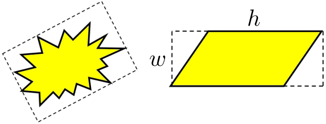
$$\text{Conc.}(S) = \frac{P(S)}{P(H(S))}$$



- **Rectangularity:**

Ratio of the width w and height h of a rotated minimum area bounding rectangle R_θ

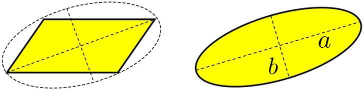
$$\text{Rect.}(S) = \max_\theta \frac{w}{h} \quad \text{with} \quad w > h$$



- **Eccentricity/Elongation:**

Ratio of the major a and minor b axis of a rotated minimum area bounding ellipse e_θ

$$\text{Ecc.}(S) = \max_\theta \frac{a}{b} \quad \text{with} \quad a > b$$



Moments

With f the binary image (0=background, 1=shape) of size $h \times w$

- **Space moments:**

$$M_{pq}(S) = \frac{1}{hw} \sum_{x=1}^{w} \sum_{y=1}^{h} x^p y^q f(x,y) \qquad \text{at order} \quad p+q$$

  – Order 0: Surface

$$M_{00} = \frac{A(S)}{hw}$$

  – Order 1: Gravity center

$$\begin{cases} \bar{x} = \frac{M_{10}}{M_{00}} \\ \bar{y} = \frac{M_{01}}{M_{00}} \end{cases}$$

Moments

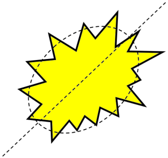
- **Centered moments:** (translation invariant)

$$\mu_{pq}(S) = \frac{1}{hw}\sum_{x=1}^{w}\sum_{y=1}^{h}(x-\bar{x})^p(y-\bar{y})^q f(x,y) \qquad \text{at order} \quad p+q$$

  – Order 2: Inertia matrix

$$\theta = \frac{1}{2}\arctan\frac{2\mu_{11}}{\mu_{20}-\mu_{02}} \quad \text{orientation}$$

$$e = \frac{\sqrt{(\mu_{20}-\mu_{02})^2+4\mu_{11}}}{\mu_{20}+\mu_{02}} \quad \text{eccentricity}$$



- **Normalized moments:** (invariant to scale)

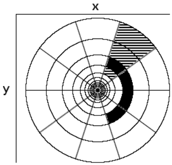
$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \qquad \text{with} \quad \gamma = \frac{p+q}{2}+1 \quad \text{for} \quad p+q \geq 2$$

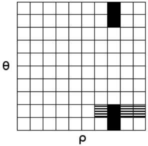
- **Hu moments:** Combination of normalized moments of orders 2 and 3

Other Moments

- **Other base moments:**
  - Legendre polynomial
  - Zernike polynomial
  - Angular Radial Transform (MPEG-7)

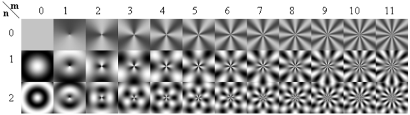
$$F_{nm} = \int_{\theta=0}^{2\pi} \int_{\rho=0}^{1} V_{nm}(\rho,\theta) f(\rho,\theta) \rho d\rho d\theta$$





$$V_{nm}(\rho,\theta) = A_m(\theta) R_n(\rho) \longrightarrow R_n(\rho) = \begin{cases} 1 & n = 0 \\ 2cos(\pi n \rho) & n \neq 0 \end{cases}$$

$$A_m(\theta) = \frac{1}{2\pi} e^{jm\theta}$$



Shape descriptors

Question: Possible definition of a *regularity* criteria for a shape?

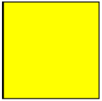
What is a regular shape? What are its properties? How to measure them?

Shape descriptors

Question: Possible definition of a *regularity* criteria for a shape?

What is a regular shape? What are its properties? How to measure them?

Regular shape?

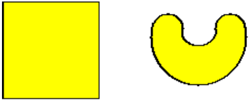


Shape descriptors

Question: Possible definition of a *regularity* criteria for a shape?

What is a regular shape? What are its properties? How to measure them?

Regular shape?

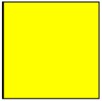


Shape descriptors

Question: Possible definition of a *regularity* criteria for a shape?

What is a regular shape? What are its properties? How to measure them?

Regular shape?



Convex?

Shape descriptors

Question: Possible definition of a *regularity* criteria for a shape?

What is a regular shape? What are its properties? How to measure them?

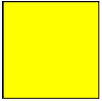
Regular shape? Convex?

Shape descriptors

Question: Possible definition of a *regularity* criteria for a shape?

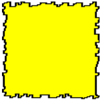
What is a regular shape? What are its properties? How to measure them?

Regular shape?



Convex?

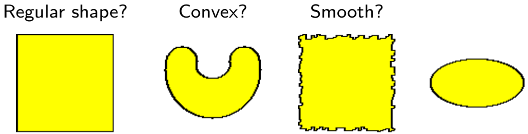
Smooth?



Shape descriptors

Question: Possible definition of a *regularity* criteria for a shape?

What is a regular shape? What are its properties? How to measure them?

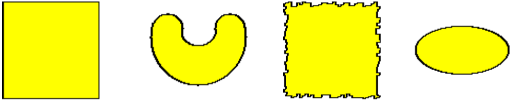


Shape descriptors

Question: Possible definition of a *regularity* criteria for a shape?

What is a regular shape? What are its properties? How to measure them?

Regular shape? Convex? Smooth? Balanced?

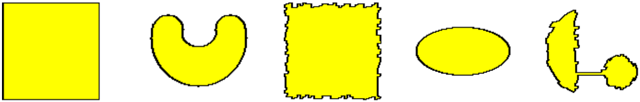


Shape descriptors

Question: Possible definition of a *regularity* criteria for a shape?

What is a regular shape? What are its properties? How to measure them?

Regular shape? Convex? Smooth? Balanced? ?

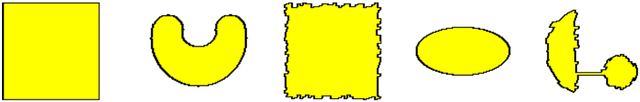


Convexity
Smoothness
Balance

Shape descriptors

Question: Possible definition of a *regularity* criteria for a shape?

What is a regular shape? What are its properties? How to measure them?



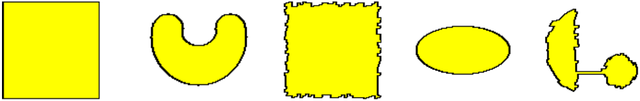
	Regular shape?	Convex?	Smooth?	Balanced?	?
Convexity	1.00	**0.80**	0.93	0.99	**0.54**
Smoothness					
Balance					

- **Convexity:** Overlap with the convex hull: Conv.$(S) = \frac{A(S)}{A(H(S))}$

Shape descriptors

Question: Possible definition of a *regularity* criteria for a shape?

What is a regular shape? What are its properties? How to measure them?



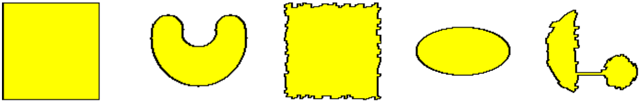
	Regular shape?	Convex?	Smooth?	Balanced?	?
Convexity	1.00	**0.80**	0.93	0.99	**0.54**
Smoothness	1.00	0.87	**0.77**	1.00	**0.66**
Balance					

- **Convexity:** Overlap with the convex hull: $\text{Conv.}(S) = \frac{A(S)}{A(H(S))}$
- **Smoothness:** Ratio with the convex hull perimeter: $\frac{1}{\text{Conc.}(S)} = \frac{P(H(S))}{P(S)}$

Shape descriptors

Question: Possible definition of a *regularity* criteria for a shape?

What is a regular shape? What are its properties? How to measure them?



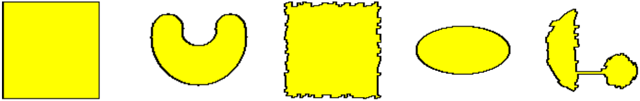
	Regular shape?	Convex?	Smooth?	Balanced?	?
Convexity	1.00	**0.80**	0.93	0.99	**0.54**
Smoothness	1.00	0.87	**0.77**	1.00	**0.66**
Balance	1.00	0.81	1.00	**0.72**	**0.86**

- **Convexity:** Overlap with the convex hull: $\text{Conv.}(S) = \frac{A(S)}{A(H(S))}$
- **Smoothness:** Ratio with the convex hull perimeter: $\frac{1}{\text{Conc.}(S)} = \frac{P(H(S))}{P(S)}$
- **Balance:** Ratio of pixel position variances: $\sqrt{\frac{\min(\sigma_x, \sigma_y)}{\max(\sigma_x, \sigma_y)}}$

Shape descriptors

Question: Possible definition of a *regularity* criteria for a shape?

What is a regular shape? What are its properties? How to measure them?



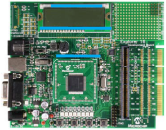
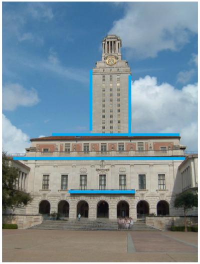
	Regular shape?	Convex?	Smooth?	Balanced?	?
Convexity	1.00	0.80	0.93	0.99	0.54
Smoothness	1.00	0.87	0.77	1.00	0.66
Balance	1.00	0.81	1.00	0.72	0.86
SRC	1.00	0.56	0.72	0.71	0.31

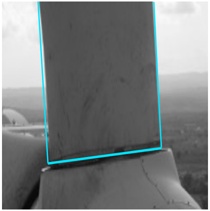
- **Convexity:** Overlap with the convex hull: $\text{Conv.}(S) = \frac{A(S)}{A(H(S))}$
- **Smoothness:** Ratio with the convex hull perimeter: $\frac{1}{\text{Conc.}(S)} = \frac{P(H(S))}{P(S)}$
- **Balance:** Ratio of pixel position variances: $\sqrt{\frac{\min(\sigma_x, \sigma_y)}{\max(\sigma_x, \sigma_y)}}$
- **Shape Regularity Criteria:** (SRC) Multiplication of the 3 previous criteria

Hough Transform (HT)*

Definition: Detection of lines (or other shapes) in a contour image

Motivation: Many objects are characterized by their edges

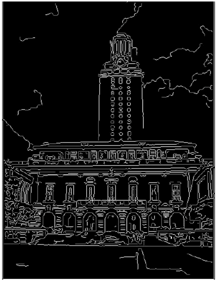


*Hough, Machine Analysis of Bubble Chamber Pictures, Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Contour image

Considers a binary **contour image**

Can be easily obtained with filter detection methods (Canny, Sobel) or more advanced techniques including supervised and deep learning

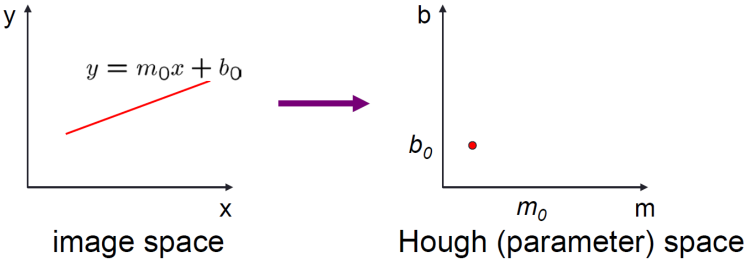
Lines can be difficultly extracted from contour image due to:

- Partial information: missing points in a line
- Unnecessary information: contour points not belonging to a line
- The presence of noise

Principle

Idea: Searching the information in the line parameter space **(m,b)**

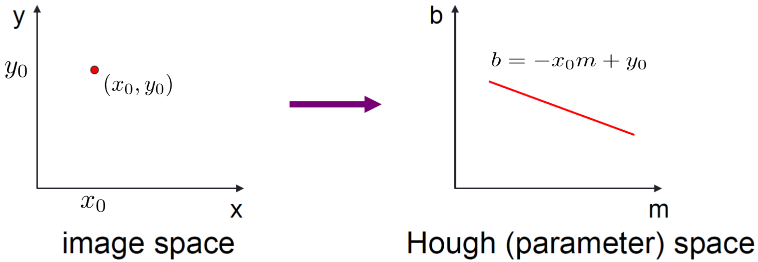
A line in the image space corresponds to a point in the Hough space



Principle

Idea: Searching the information in the line parameter space **(m,b)**

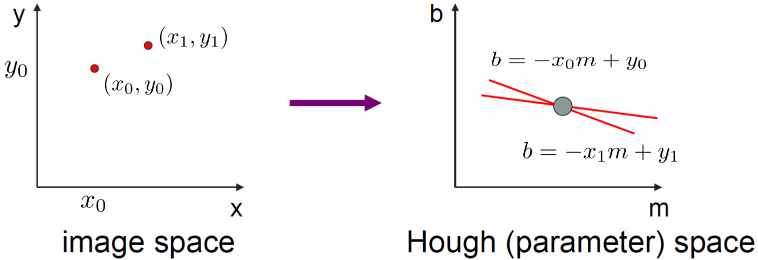
A point in the image space corresponds to a line in the Hough space



Principle

Idea: Searching the information in the line parameter space **(m,b)**

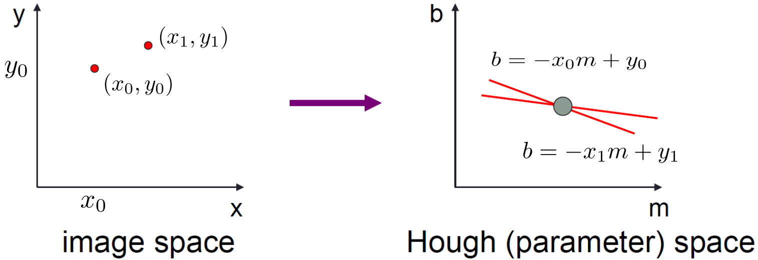
The line passing by (x_0, y_0) and (x_1, y_1) corresponds to the intersection between the lines $-x_0 m + y_0$ and $-x_1 m + y_1$



Principle

Idea: Searching the information in the line parameter space **(m,b)**

The line passing by (x_0, y_0) and (x_1, y_1) corresponds to the intersection between the lines $-x_0 m + y_0$ and $-x_1 m + y_1$



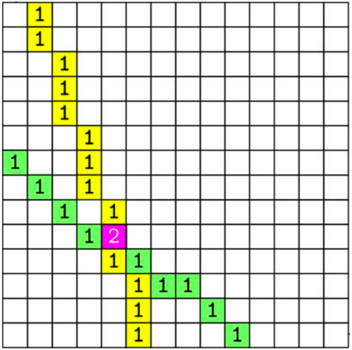
→ Adding the lines in the Hough space for each point of the contour map.

→ The most probable lines in the image are defined by the highest values

Principle

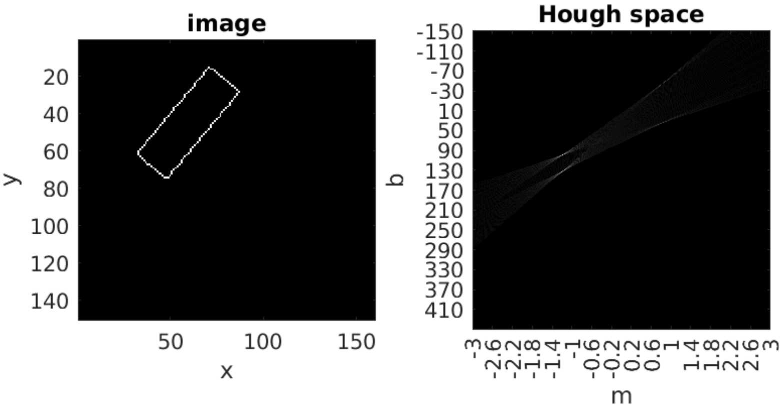
Accumulation of the lines in the Hough space

- The space **(m,b)** needs to be discretized
  - Trade-off between accuracy and over-detection



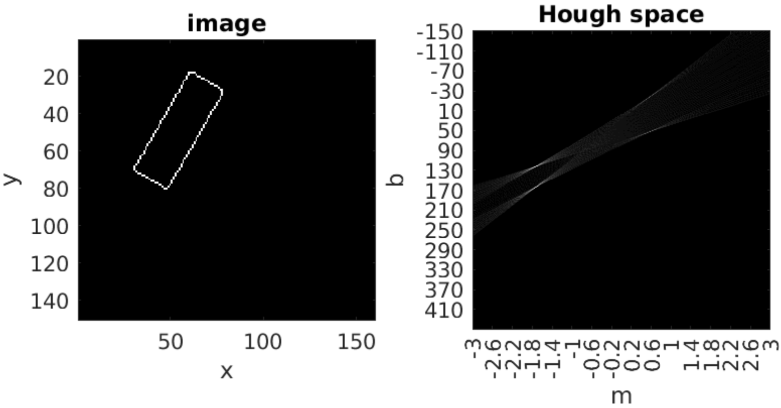
Principle

Problem: The slope can go to the infinity when the line is vertical ($y = mx + b$)



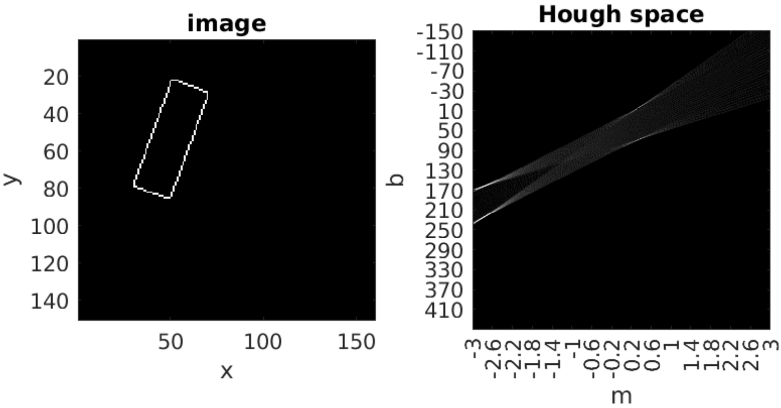
Principle

Problem: The slope can go to the infinity when the line is vertical ($y = mx + b$)



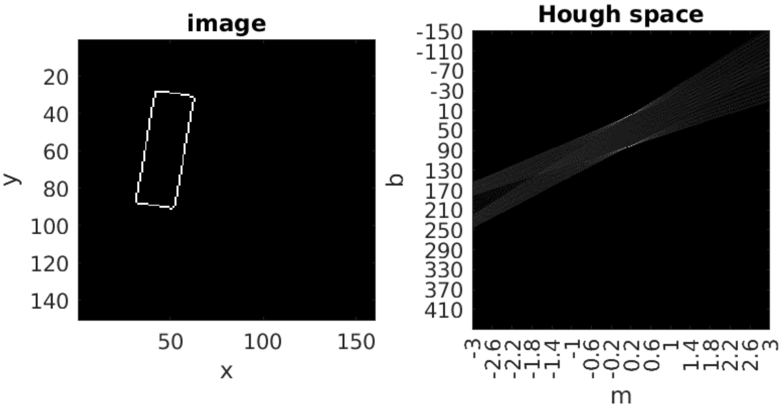
Principle

Problem: The slope can go to the infinity when the line is vertical ($y = mx + b$)



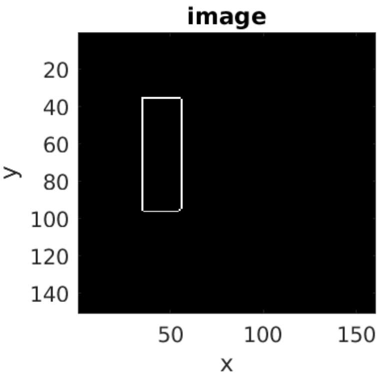
Principle

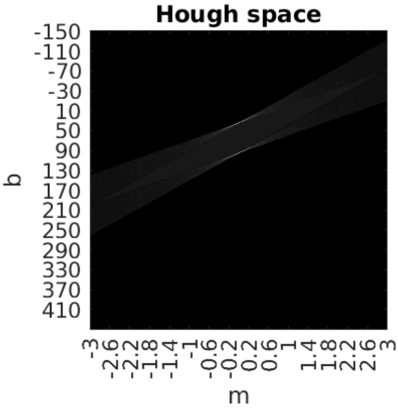
Problem: The slope can go to the infinity when the line is vertical ($y = mx + b$)



Principle

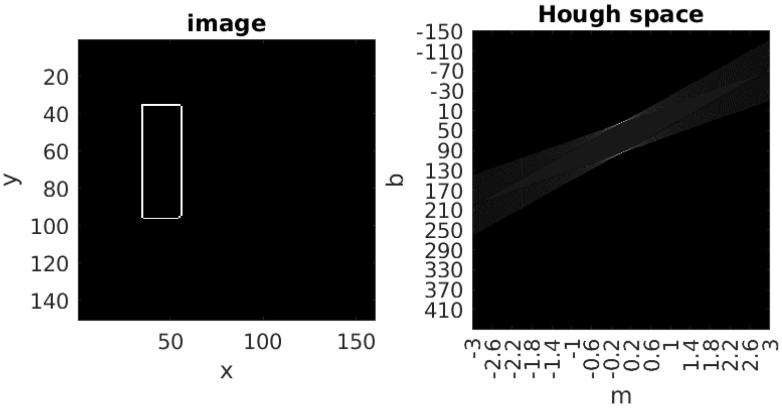
Problem: The slope can go to the infinity when the line is vertical ($y = mx + b$)





Principle

Problem: The slope can go to the infinity when the line is vertical ($y = mx + b$)

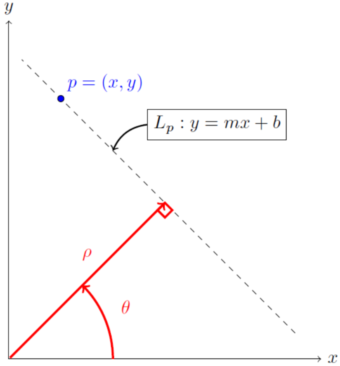


→ Polar coordinates (ρ, θ)

Polar coordinates*

In practice, the bounded polar space $(\boldsymbol{\rho}, \boldsymbol{\theta})$ is used

$$y = mx + b \Leftrightarrow xcos(\theta) + ysin(\theta)$$



$\theta \in [0, 180]$

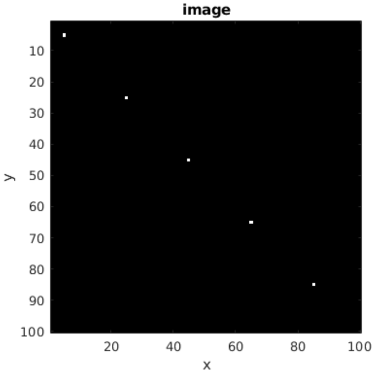
$\rho \in [-R, R]$ with $R = \sqrt{h^2 + w^2}$

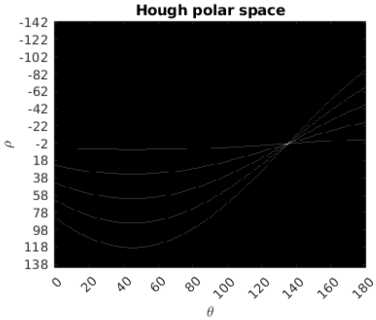
*Duda et al., Use of the Hough Transformation to Detect Lines and Curves in Pictures, Comm. ACM, 1972

Polar coordinates

A point in **(x,y)** becomes a sinusoid in $(\boldsymbol{\rho}, \boldsymbol{\theta})$

A line in **(x,y)** is still a point in $(\boldsymbol{\rho}, \boldsymbol{\theta})$

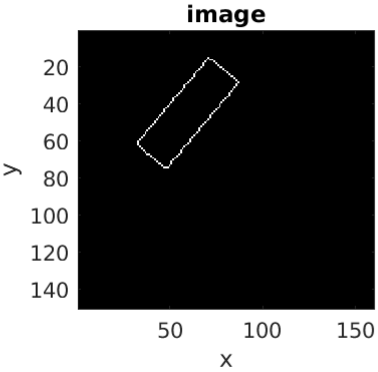


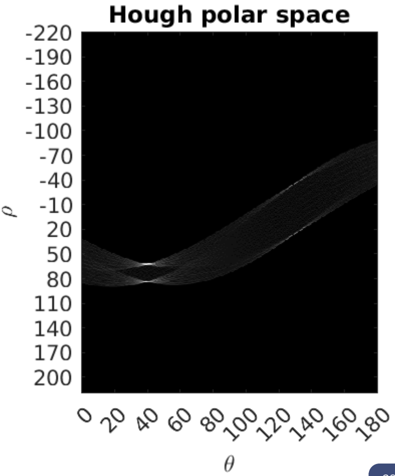


Polar coordinates

A point in **(x,y)** becomes a sinusoid in $(\boldsymbol{\rho}, \boldsymbol{\theta})$

A line in **(x,y)** is still a point in $(\boldsymbol{\rho}, \boldsymbol{\theta})$

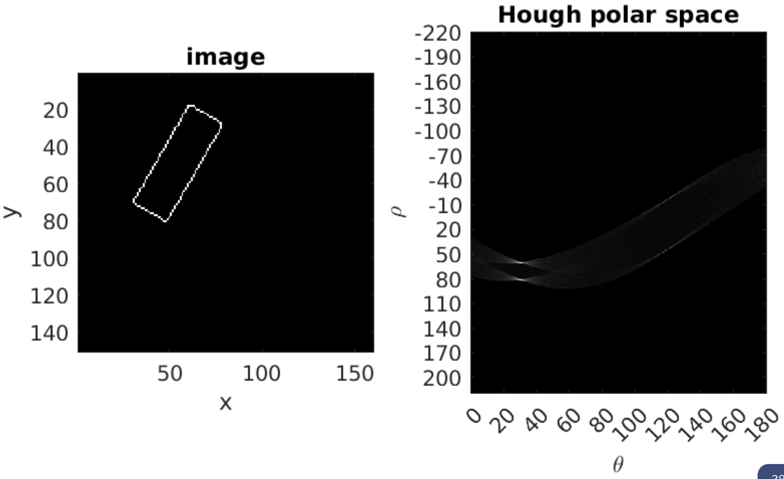




Polar coordinates

A point in **(x,y)** becomes a sinusoid in (ρ, θ)

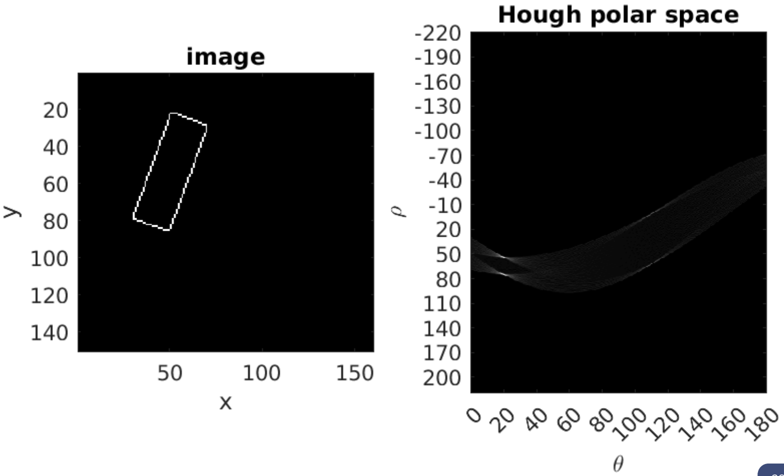
A line in **(x,y)** is still a point in (ρ, θ)



Polar coordinates

A point in **(x,y)** becomes a sinusoid in $(\boldsymbol{\rho}, \boldsymbol{\theta})$

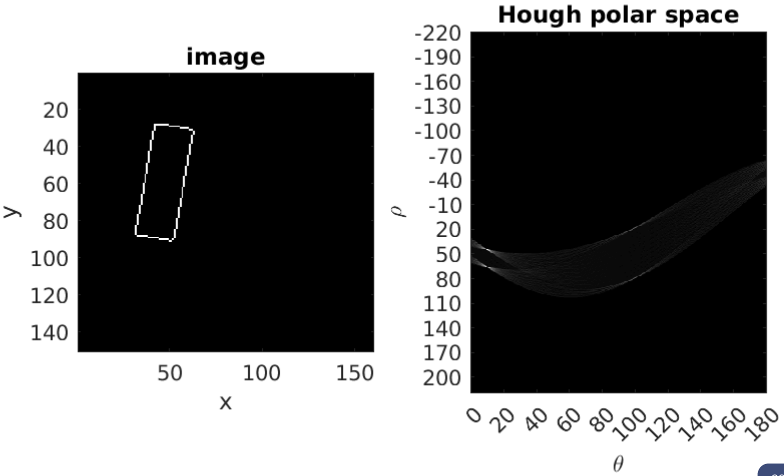
A line in **(x,y)** is still a point in $(\boldsymbol{\rho}, \boldsymbol{\theta})$



Polar coordinates

A point in **(x,y)** becomes a sinusoid in $(\boldsymbol{\rho}, \boldsymbol{\theta})$

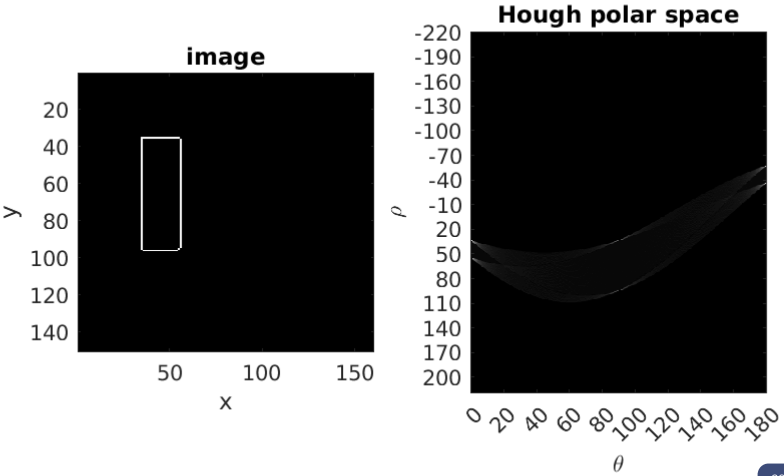
A line in **(x,y)** is still a point in $(\boldsymbol{\rho}, \boldsymbol{\theta})$



Polar coordinates

A point in **(x,y)** becomes a sinusoid in $(\boldsymbol{\rho}, \boldsymbol{\theta})$

A line in **(x,y)** is still a point in $(\boldsymbol{\rho}, \boldsymbol{\theta})$



Extension to other shapes

The same method can be applied to any curve represented by a Cartesian or parametric equation.

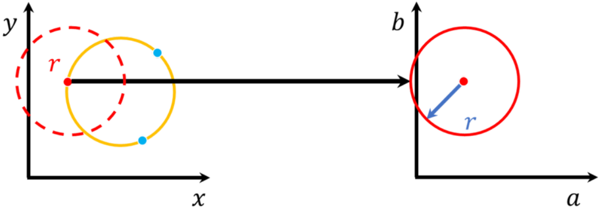
- **Hough Circle Transform (HCT):**

  Equation of a circle of center (a, b) and radius r:

  $$(x - a)^2 + (y - b)^2 = r^2$$

  If r is known, the parameter space is **(a,b)** (same dimension as **(x,y)**)

  In this space, a circle in **(x,y)** is represented by a point



Extension to other shapes

The same method can be applied to any curve represented by a Cartesian or parametric equation.

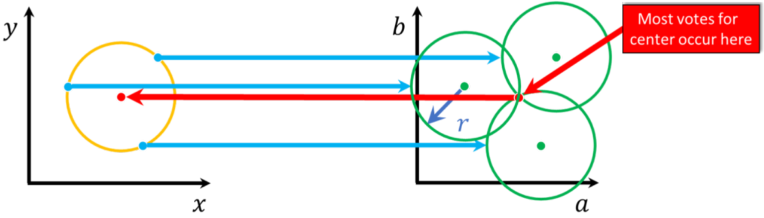
- **Hough Circle Transform (HCT):**

  Equation of a circle of center (a, b) and radius r:

  $$(x - a)^2 + (y - b)^2 = r^2$$

  If r is known, the parameter space is **(a,b)** (same dimension as **(x,y)**)

  In this space, a circle in **(x,y)** is represented by a point



Extension to other shapes

The same method can be applied to any curve represented by a Cartesian or parametric equation.

- **Hough Circle Transform (HCT):**

  Equation of a circle of center (a, b) and radius r:

  $$(x-a)^2 + (y-b)^2 = r^2$$

  If r is known, the parameter space is **(a,b)** (same dimension as **(x,y)**)

  In this space, a circle in **(x,y)** is represented by a point

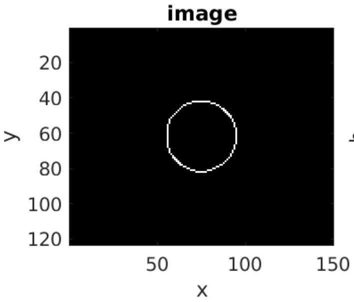
  Method: For each pixel of the contour map that is detected as contour point (x, y), consider each centrer location a within $[x-r, x+r]$ and accumulates in the accumulation matrix a value at the two locations $(a, y - \sqrt{r^2 - (x-a)^2})$ and $(a, y + \sqrt{r^2 - (x-a)^2})$ that intersect the point (x, y)

  If r is not know, the parameter space becomes **(a,b,r)**

  Applications: Detection of faces, road signs, default or impacts on lenses, aneurysms detection on angiograms, ...

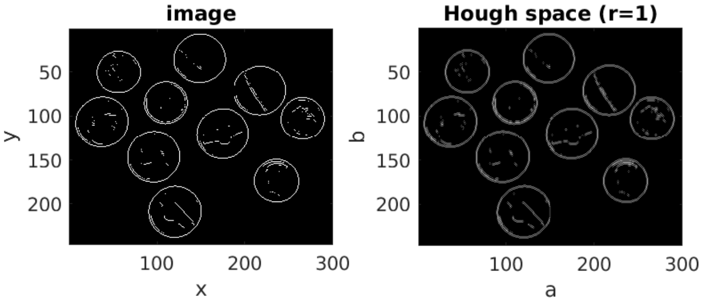
Extension to other shapes

Result examples:



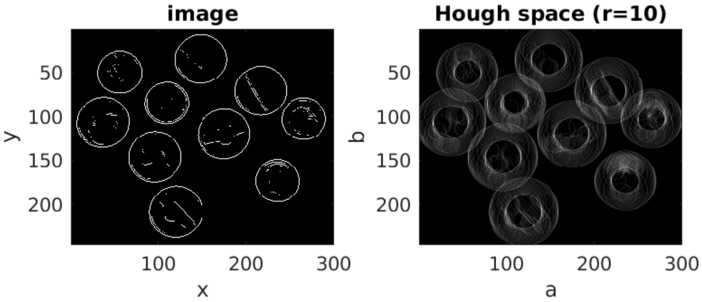
Extension to other shapes

Result examples:



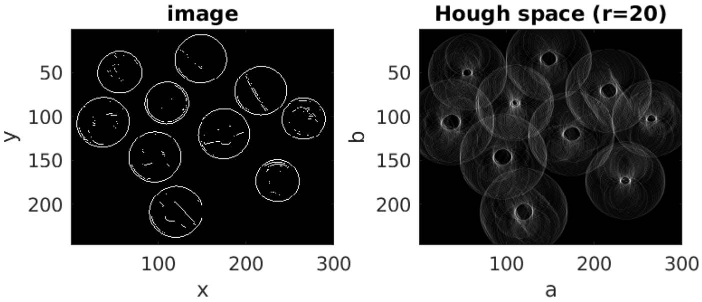
Extension to other shapes

Result examples:



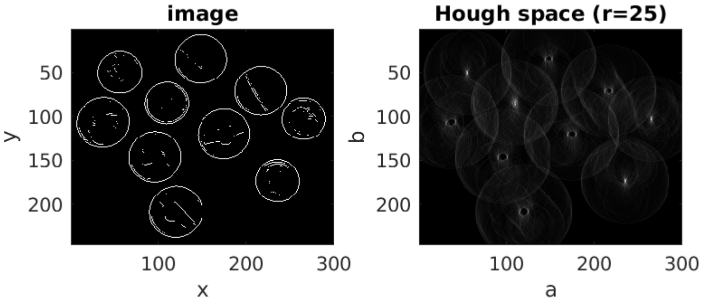
Extension to other shapes

Result examples:



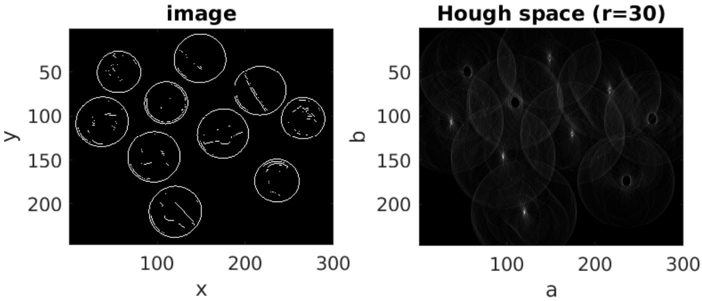
Extension to other shapes

Result examples:



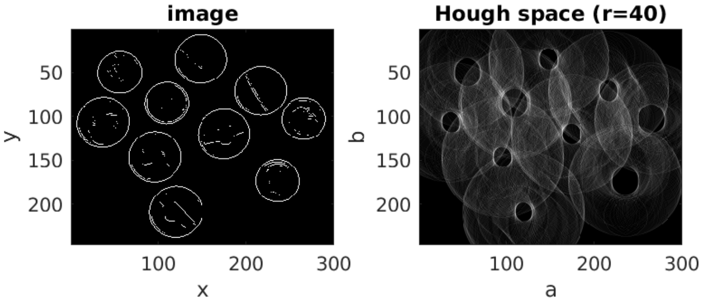
Extension to other shapes

Result examples:



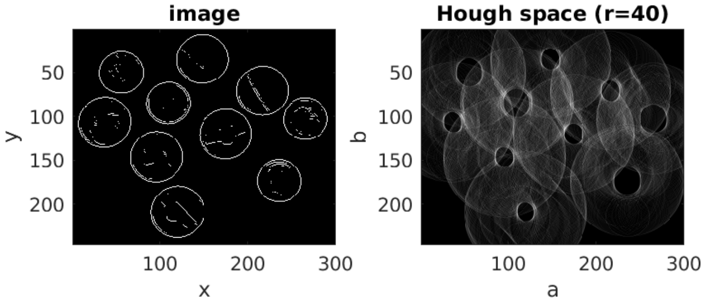
Extension to other shapes

Result examples:



Extension to other shapes

Result examples:



- **Ellipses:**
  5-parameter space (x_0, y_0, a, b, θ)
- **Planes in 3D point clouds:** (Limberger et al., 2015)
  3-parameter space (θ, ϕ, ρ)
- **Generalized Hough Transform (GHT):** (Ballard et al., 1981)
  Generalization to any shape described by contours points

Summary

- Fast method to detect simple shapes such as lines, circles, ellipses in contour images
- Use of polar coordinates to have a bounded parameter space
- Can be used to detect any shape with the Generalized Hough Transform

- **Limitations:**
  - The quantization of the parameter space may cause under or over-detection
  - Can give misleading results when objects happen to be aligned by chance.
  - Detected lines are infinite lines described by their (m,c) values, rather than finite lines with defined end points.
  - The computational cost can be high for a large parameter space

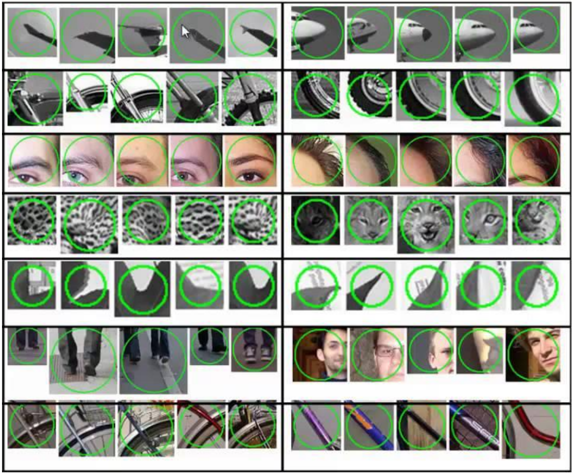
Practical n°1

- **Data:** Road sign images

- **Hough Circle Transform (HCT)**
  - Compute a contour map (`skimage.feature.canny`)
  - Implement the HCT
  - Filter the Hough map with a non-maximum suppression: only keep maximum values in a $l \times l$ region (`np.ravel, np.argmax`)
  - Find a way to get the inner circle surrounding the limitation number (`np.ravel, np.argsort, np.unravel_index`)
  - Display it on the input image (`np.meshgrid`)

What is a pattern?

Definition: Region + {Properties} (color, orientation, repetition)
An object or object region with specific properties

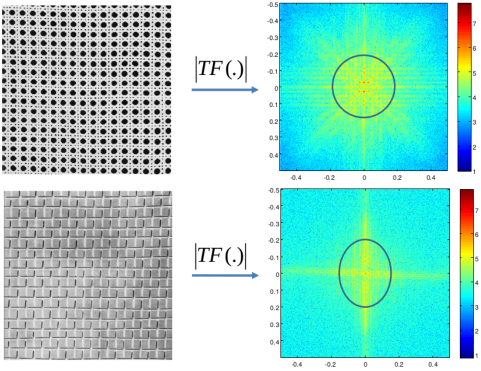


Texture

Definition: Repetitive spatial arrangement of pixels

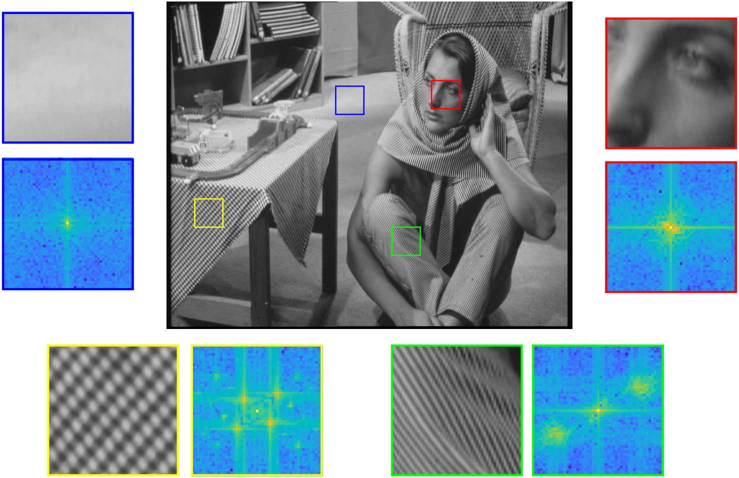
A term strongly connected to Pattern

Frequential space (Fourier Transform) well suited to exhibit texture information
$\rightarrow$ Dirac corresponding to spatially repetitive patterns



Texture

- Textures are naturally present in images
  - But the frequential space may be insufficient to describe complex objects



Descriptor Properties

Definition: A set of features describing (a region of) the image

A “good” descriptor should provide: “similar” values for “similar” patterns
“different” values for “different” patterns

Properties:

- Rotation invariance: be able to identify the object, regardless of its orientation
- Translation invariance: even if the object is moved to a different location, it should be detected
- Scale invariance: should work even if the image is zoomed in or out
- Illumination invariance: should work even if there is change in brightness and contrast in the image
- Robustness to noise, deformations, occlusions
- Need for parameter tuning
- Time to compute
- Size
- ...

What kind of descriptor?

Dense (all pixels)

Dense features

- **Intensity/color**
- **Convolutions**
- **Local Binary Patterns (LBP)**
- Mean LBP
- Multi-Block LBP
- LBP Histogram Fourier
- ...

Local (keypoints)

Keypoint detectors

- **Harris corners**
- Shi-Tomasi corners
- FAST corners
- **SIFT keypoints**
- SURF keypoints
- ...

Local descriptors

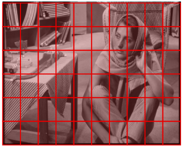
- **SIFT**
- SURF
- BRIEF
- ORB
- ...

Global (all pixels)

Global descriptors

- **Co-occurrence matrix**
- **Fourier transform**
- **Histogram of dense features**
- ...

Block-wise (all pixels)



Block-wise descriptors

- **Block-wise histograms of dense features**
- **Histogram of Oriented Gradients (HOG)**
- Gradient Field HOG
- ...

Dense features: Image transformations to exhibit features for each pixel

Local descriptor on Keypoints: Generally for image matching

Global descriptor: Single descriptor for the whole image (loss of spatiality)

Block-wise descriptor: Image divided into blocks to extract features $\rightarrow$ spatiality

Dense Feature Extraction

Definition: Each pixel is described by a set of features

Semi-dense: For each pixel at a lower scale (1/2, 1/4, ...)

- **Intensities/colors** of the initial image

Intensity

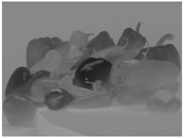
R

G

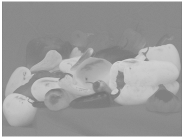
B

- **Color spaces** by linear and non-linear transformations (YUV, YCbCr, ...)

Y



Cb



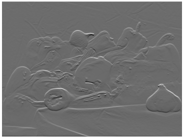
Cr

Dense Feature Extraction

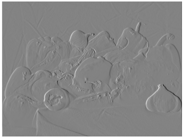
- **Result of convolutional filtering**

  Edge detection (Sobel, Canny, ...)

Intensity

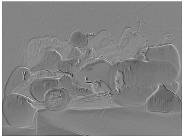


Sobel x

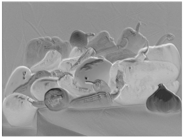


Sobel y

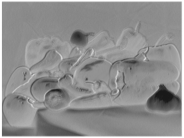
  Learned filters (Deep learning)



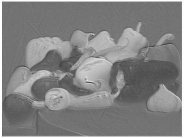
Filtering 1



Filtering 2



Filtering 3



Filtering 4

Local Binary Patterns (LBP)*

Definition: Binary code corresponding to differences with neighboring pixels

Motivation: Comparing the intensity of a pixel with the ones of its neighbors extracts a regular pattern information in the image, *i.e.*, a texture information.

→ Efficient descriptor for texture classification and also face recognition

→ Generates a map of the image size → histogram → classification system

Image

LBP map

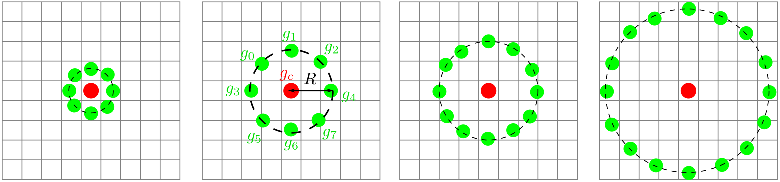
*Harwood et al., Texture classication a by center-symmetric auto-correlation, using Kullback discrimination of distributions, Technical Reports, 1993

Neighborhood

1) Define the neighborhood:

Originally a 3x3 patch (standard 8-neighborhood) was considered

Generalization to a circularly symmetric neighborhood of radius R^*



$P = 8, R = 1$ — $P = 8, R = 2$ — $P = 12, R = 2$ — $P = 12, R = 3$

→ Each pixel g_c has P neighbors $\{g_0, g_1, ..., g_{P-1}\}$ (ordered by convention)

→ The case $P = 8,\ R = 1$ corresponds to the standard square 8-neighborhood

→ If $R > 1$, the neighborhood can be computed by:

- Nearest neighbor approach: selecting the closest pixel
- Bilinear interpolation: averaging the 4 closest pixels

*Ojala et al., Multiresolution gray-scale and rotation invariant texture classification with local binary patterns, PAMI, 2002

Local Binary Patterns Coding

2) Compute the LBP:

For each pixel g_c having P neighbors $\{g_0, g_1, ..., g_{P-1}\}$

1. Compute the intensity difference:

$$\{g_0 - g_c, g_1 - g_c, ..., g_{P-1} - g_c\}$$

2. Apply the sign function

$$\delta(x) = \begin{cases} 1 & \text{if} \quad x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

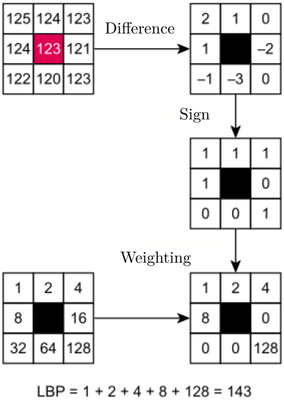
3. Weight according to binary coding

$$\text{LBP}(g_c) = \sum_{p=0}^{P-1} 2^p \delta(g_p - g_c)$$

$\rightarrow$ This gives a unique code for each pattern

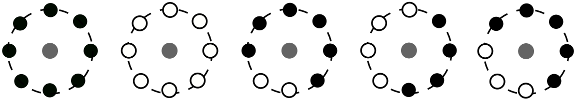
$\rightarrow$ 2^P different codes, $P = 8$: uint8 coding

Case $P = 8$



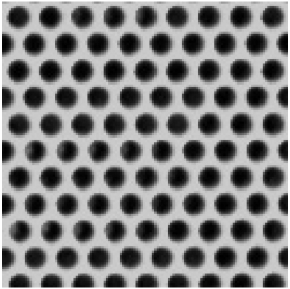
Local Binary Patterns Coding

Examples of patterns detected by the LBP:

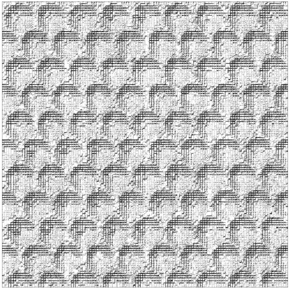


Spot Spot/flat Line end Edge Corner

LBP value for each pixel:



Image



LBP map

Extensions

The LBP method has been extended over the years:

- Improved LBP (M-LBP): (Jin et al., 2004)
  Comparison of all the pixels (including the central pixel) with the mean intensity of the neighborhood → more discriminative power
- Multi-Block LBP (MB-LBP): (Zhang et al., 2007)
  Capture micro- and macro- structure information by comparing average intensities of neighboring sub-regions (muti-scale) → more efficient
- 3D LBP: (Fehr et al., 2007)
  Straightforward extension of LBP to 3D volume data
- LBP and Fourier Transform (LBP-HF): (Ahonen et al., 2009)
  Combines LBP and Discrete FT → invariance to rotation
- ...

LBP Summary

- Simple coding of texure pattern with many implementation variations
- Dense feature: a descriptor is computed for each pixel
- Efficient application to texture classification and face recognition

Advantages

- Robust to illumination variations
- Invariant to rotation and scale
- Computationally efficient $\rightarrow$ interesting for large datasets and real-time

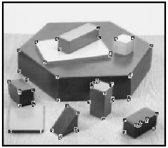
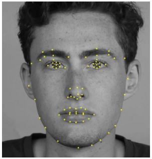
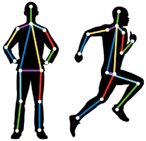
Disadvantages

- Sensitive to noise
- May fail to capture more global texture information
- Invariant to rotation (if rotation is important to charaterize textures)
- Does not explicitly capture color information

Keypoints

Definition: Locations of pixels containing relevant information for a targeted task

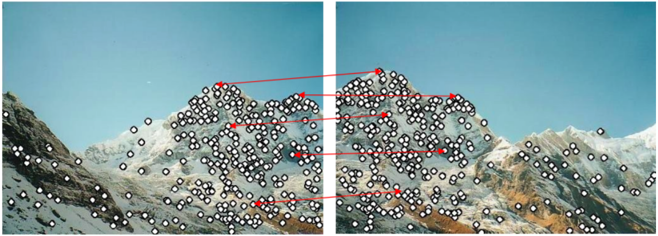
- Points with contrast variations in their neighborhood (texture)
- Corners to detect the edges of structures
- Body parts (arm, shoulder, knees) for pose estimation
- Face parts (eyebrow, lips, nose) for face recognition
- Specific fingerprint locations for identification
- ...


Keypoints

Generally used for image matching/registration:

$\rightarrow$ Keypoints = discriminant image locations that can be found in another point of view



How to detect such keypoints?

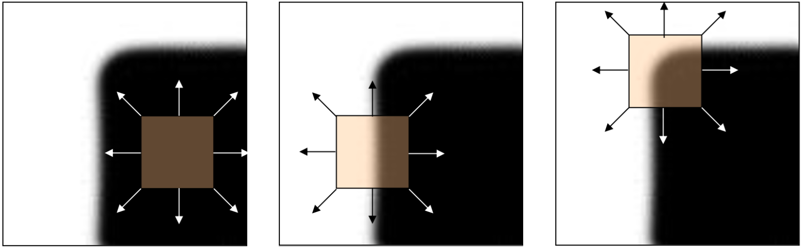
- Corner detection: Harris methods
- Highly contrasted local extremum: Scale Invariant Feature Tranform (SIFT)

Harris corner detector

Principle: Detect the image corners using vertical and horizontal gradients

How to characterize a corner?

Homogeneous region:	no change in intensity
Contour:	no change along the contour
Corner:	significant change in at least 2 directions



Homogeneous region — Contour — Corner

Harris corner detector

This information can be found in the gradients distribution

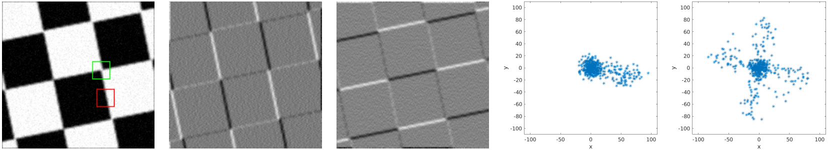


Image | $\mathbf{I_x}$ | $\mathbf{I_y}$ | Edge | Corner

Method:

1)- Compute the gradient maps $\mathbf{I_x}$ and $\mathbf{I_y}$ (Sobel, Gaussian derivates, ...)

2)- For each region $l \times l$ around a pixel (x, y), compute the covariance matrix of the gradient values $\mathbf{x} = \mathbf{I_x}(x, y)^l$ and $\mathbf{y} = \mathbf{I_y}(x, y)^l$

$$\mathbf{C}(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \text{Cov}(\mathbf{x}, \mathbf{x}) & \text{Cov}(\mathbf{x}, \mathbf{y}) \\ \text{Cov}(\mathbf{y}, \mathbf{x}) & \text{Cov}(\mathbf{y}, \mathbf{y}) \end{bmatrix} = \begin{bmatrix} \sigma(\mathbf{x})^2 & \text{Cov}(\mathbf{x}, \mathbf{y}) \\ \text{Cov}(\mathbf{y}, \mathbf{x}) & \sigma(\mathbf{y})^2 \end{bmatrix}$$

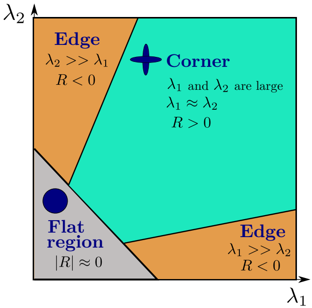
$\rightarrow$ The eigen values (λ_1, λ_2) give information about the variance and correlation between $\mathbf{x}$ and $\mathbf{y}$ (cf. PCA)

Harris corner detector

3)- Measure the dispersion by computing the map:

$$\mathbf{R}(x,y) = \det(\mathbf{C}) - k\text{tr}(\mathbf{C})^2$$
$$= \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2$$

with $k \in [0.04, 0.06]$ empirically set, λ_1, λ_2 the eigen values of $\mathbf{C}$



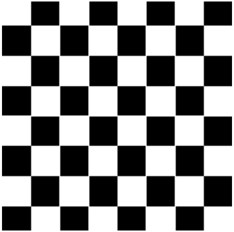
$\rightarrow$ The highest values of $\mathbf{R}$ correspond to the most probable corners

4)- To avoid over-detection: only keep the maximum values in a 3×3 region Non-Maximum Suppression (NMS)

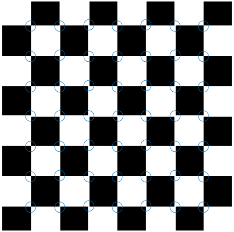
5) Corner selection:
Select the top-N values or all locations where $\mathbf{R} >$ threshold

Harris corner detector

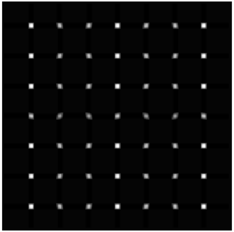
Examples: with $l = 3$ and corners such as $\mathbf{R} > 0.001$



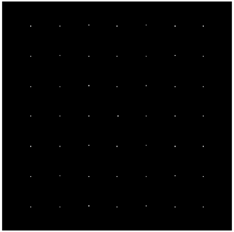
Image



Corner detection



$\mathbf{R}$



$\mathbf{R}$ after NMS

Harris corner detector

Examples: with $l = 3$ and top-50 corners

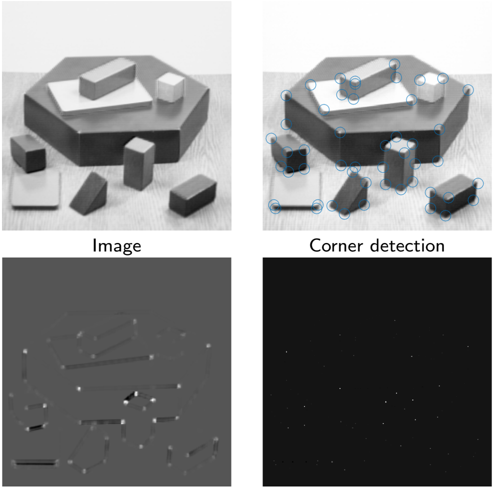


Image — Corner detection

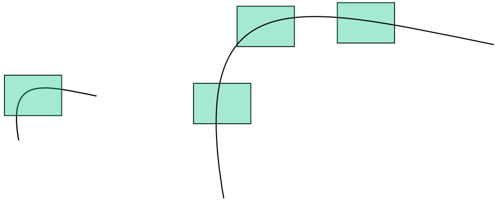
$\mathbf{R}$ — $\mathbf{R}$ after NMS

Harris corner detector

Properties:

- Invariant to rotation
- Non-invariant to scale !

For example, a corner can become an edge when the image is scaled but the detector is operating over the same window size.



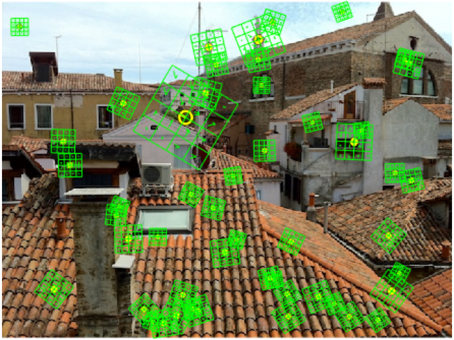
→ Scale-invariant Feature Transform (SIFT) keypoint detector

Scale-Invariant Feature Transform (SIFT)*

Definition: Highly contrasted local extremum keypoint detection and description based on gradient orientations

Motivation: Provide for an image, a robust set of rotation and scale-invariant keypoints, contrary to Harris detector, also described by a rotation and scale-invariant descriptor

The SIFT method = **SIFT keypoints** (+ SIFT descriptor)

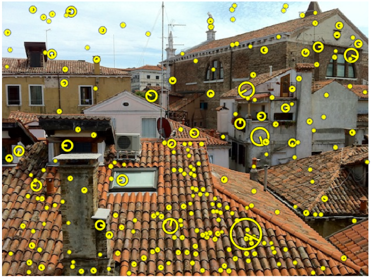


SIFT Keypoints

Idea: Create a *scale space* with the image resized at lower scales and blurred at different levels to detect significant corners and patterns at different resolutions

Keypoint detection method:

1) Scale space: Gaussian blurring at different scales
2) *Difference of Gaussian* (DoG): between images at the same scale
3) Detection of keypoints: extremum in the DoG space
4) Filtering of keypoints: edge and low-contrast



1) Scale space

- Different scales: The keypoints should be scale-invariant

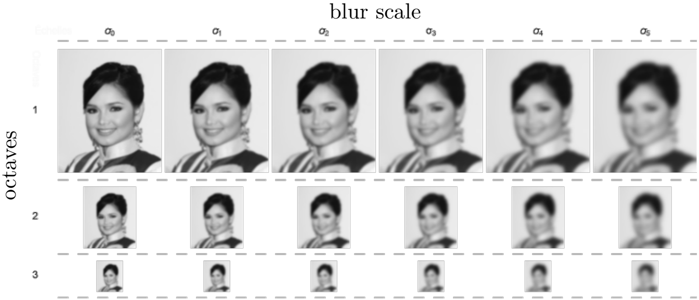
  The image $\mathbf{I}_{(h,w)}$, is resized at different scales i: $\mathbf{I}^i_{(\frac{h}{i},\frac{w}{i})}$

  The number of scales, or *octaves* (music analogy), is generally set to 4

- Gaussian blur: Cancels the impact of noise

  For a Gaussian filter $\mathbf{G}$ of variance σ: $\mathbf{L}^i(x,y,\sigma) = \mathbf{G}(x,y,\sigma) * \mathbf{I}^i(x,y)$

  Generally, 5 noise scales $\{\sigma_0, k\sigma_0, k^2\sigma_0, ...\}$ with $k = \sqrt{2}$, $\sigma_0 = 1.6$

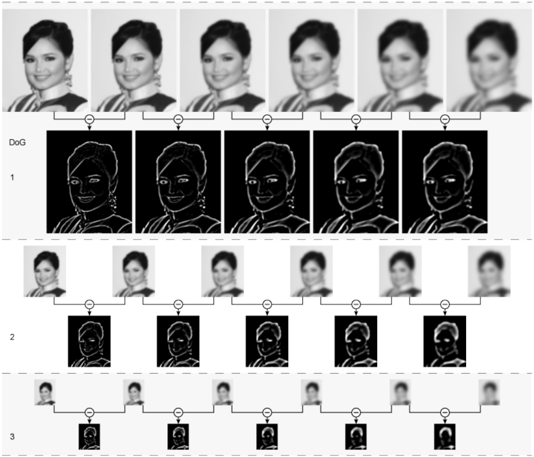


2) Difference of Gaussians (DoG)

Differences between the blurred images at the same scale:

$$\mathbf{D}^i(x, y, \sigma) = \mathbf{L}^i(x, y, k\sigma) - \mathbf{L}^i(x, y, \sigma)$$

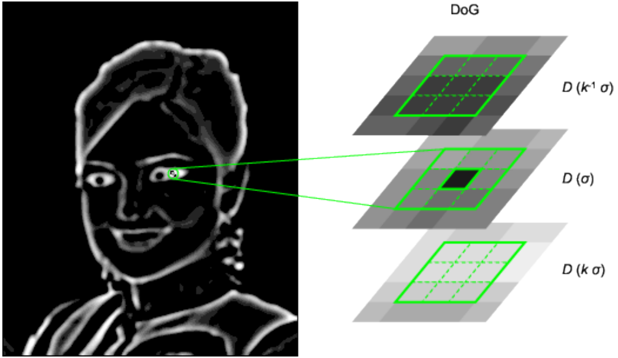
$\rightarrow$ In $\mathbf{D}^i$, only remaining objects observable between the scale $[\sigma, k\sigma]$



3) Detection of keypoints

Keypoint selection: The extremum of the DoG with respect to their immediate neighbors, i.e. on the set containing 26 other points defined by:

$$\{\mathbf{D}^i(x+\delta_x, y+\delta_y, s\sigma),\ \delta_x \in \{-1,0,1\}, \delta_y \in \{-1,0,1\}, s \in \{k^{-1}, 1, k\}\}$$



Refinement step: To better localize extremum detected at higher scales (Interpolation of derivatives)

4) Filtering of keypoints

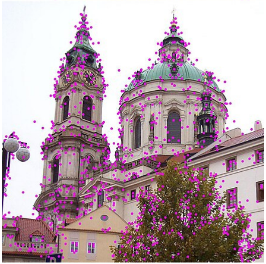
- Low contrast filtering to remove keypoints in non highly textured areas
  Thresholding of $\mathbf{D}$ values
- Edge filtering similar to Harris, on the ratio r of the eigen values of the Hessian matrix $\mathbf{H}$, with a threshold r_{th} generally set to 10:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 D}{\partial x^2} & \frac{\partial^2 D}{\partial y \partial x} \\ \frac{\partial^2 D}{\partial x \partial y} & \frac{\partial^2 D}{\partial y^2} \end{bmatrix} \qquad R = \frac{\text{tr}(\mathbf{H})^2}{\det(\mathbf{H})} = \frac{(\lambda_1 + \lambda_2)^2}{\lambda_1 \lambda_2} = \frac{(r+1)^2}{r} < \frac{(r_{th}+1)^2}{r_{th}}$$

Detected keypoints

+Low contrast filtering

+Edge filtering



Final SIFT keypoints

SIFT descriptor

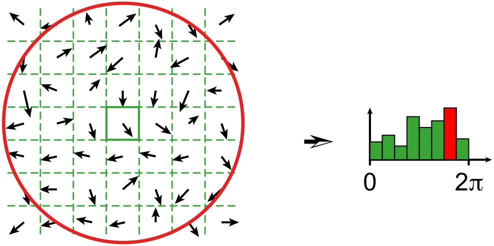
How to describe the local information at the detected keypoints?

For image matching → Scale and rotation-invariant features

1) Compute the keypoint main orientation:

Histogram (36-bins) of gradient orientations in a neighborhood

Contributions weighted by a Gaussian windows depending on $k\sigma$, the scale of the detected keypoint



Main orientation associated to the keypoint → rotation-invariance

SIFT descriptor

2) Compute the final rotation-invariant descriptor:

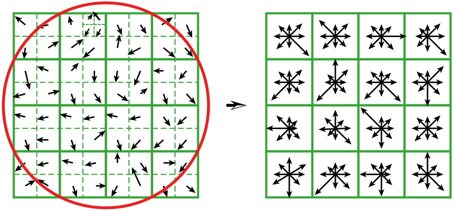
Aggregation into 8-bin histograms of gradient orientations in a 16x16 neighborhood divided into 4x4 cells, each one containing 4x4 pixels

Same Gaussian spatial weighting of contributions

Translation of minus main orientation → Rotation-invariance

Concatenation of 4x4=16 histograms → Descriptor of size 16x8=128

Normalization of the histogram ($\|.\|_2 = 1$) → Illumination invariance



→ See HOG section for more details on histogram construction

Co-occurrences matrix

Definition: Occurrences of two neighboring pixels values according to an offset

Method: For an image I of size $h \times w$

- Define a range of p values
  (the unique values of the image or the possible range, e.g., [0 255])
- Define a set of pixel offsets Δ_x, Δ_y ([0,1], [-1,1], ...)
- For each offset, compute the co-occurrence matrix :

$$C_{\Delta_x,\Delta_y}(i,j) = \sum_{x=1}^{w}\sum_{y=1}^{h}\begin{cases}1, & \text{if } I(x,y) = i \text{ and } I(x+\Delta_x, y+\Delta_y) = j\\ 0, & \text{otherwise}\end{cases}$$

1	1	5	6	8
2	3	5	7	1
4	5	7	1	2
8	5	1	2	5

Image

	1	2	3	4	5	6	7	8
1	1	2	0	0	1	0	0	0
2	0	0	1	0	1	0	0	0
3	0	0	0	0	1	0	0	0
4	0	0	0	0	1	0	0	0
5	1	0	0	0	0	1	2	0
6	0	0	0	0	0	0	0	1
7	2	0	0	0	0	0	0	0
8	0	0	0	0	1	0	0	0

Co-occurrence matrix $C_{1,0}$

Co-occurrences matrix

Examples:

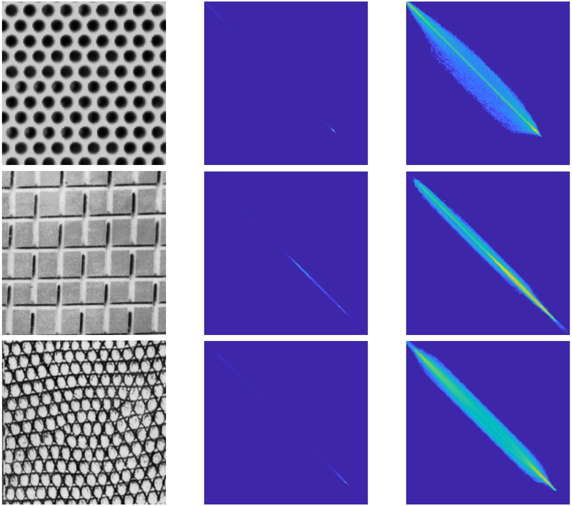
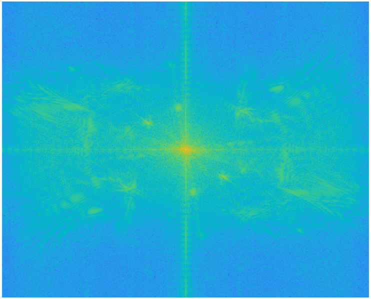


Image | Co-occurrence matrix $C_{1,0}$ | $\log(C_{1,0})$

Frequential space

Loss of spatiality but dense space of the image size

Image

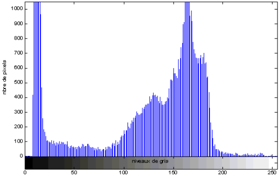


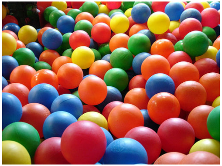
Fourier Transform (module)

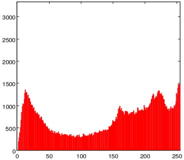
Global Histogram

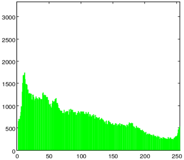
Dense features may be averaged over the whole image, *e.g.*, with a histogram, to provide a global descriptor of reduced size.

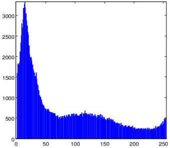
Example: Global histogram on pixel intensities/colors:







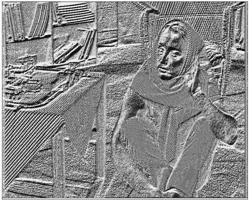




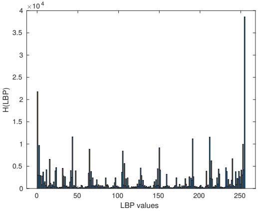
Global histogram

Histogram of the LBP map:

Image



LBP map



Global histogram

→ 2^P-bin histogram(s) vector(s) as descriptor → classification system

→ For RGB images, channels independently processed (3 maps/histograms)

Limitations: Loss of spatiality and potential blur of the information

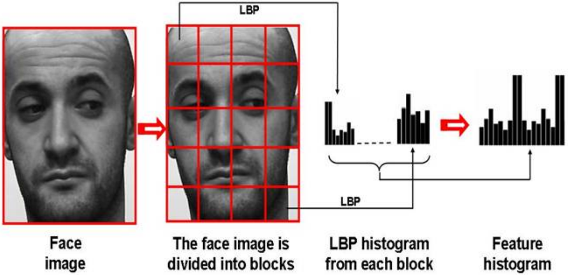
From Global to Block-wise

→ Dense features may also be synthesized in a block-wise manner

→ Preserving some spatiality and descriptive capability of the descriptor

→ Trade-off between accuracy and size/computational cost

Example: LBP for face recognition:



Note that block-wise descriptors may be simple averages into histograms but also use specific block-wise construction and normalization (cf. HOG)

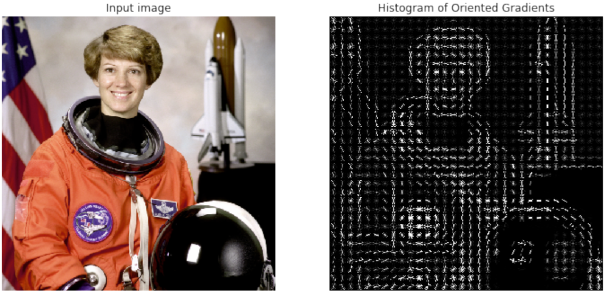
Histogram of Oriented Gradients (HOG)*

Definition: Block-wise histograms of gradient orientations

Motivation: The appearance and local shape of an object in an image can be described by the gradient intensity distribution and the direction of the edges

→ Simplification of the representation only containing important information

→ Plots of image pixel orientations and gradients on an histogram



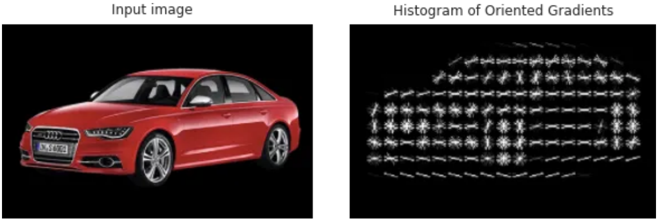
*Dalal and Triggs, Histograms of Oriented Gradients for Human Detection, CVPR, 2005

Principle

Steps to compute HOG descriptors:

1) Preprocessing: Crop or resize images (generally Xx8xYx8 pixels)
2) Compute gradient images: orientation and magnitude
3) Compute histograms of gradients: generally in 8x8 pixels cells
4) Block normalization: to normalize contrasts in blocks (generally 2x2 cells)

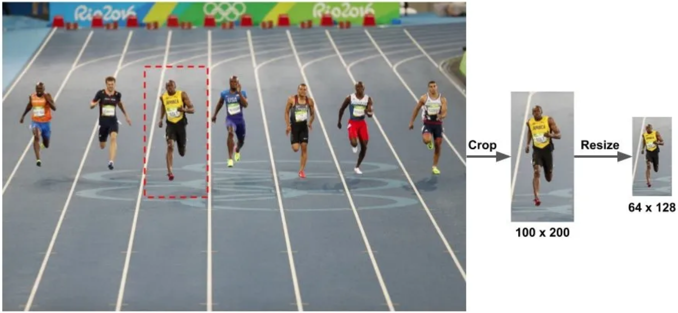
Visualization:



In each cell, histogram gradients are represented by orthogonal overlapping lines. Their intensity and direction depend on the gradient magnitude and orientation.

1) Preprocessing

- Crop and/or resize an image to fit to a standard size
- Only necessary to compare HOG descriptors of different regions or images, *e.g.*, in a classification system → input features need to have the same size
- Ideally the image dimensions are multiple of the cell (4, **8**, 16, ...) or block (8, **16**, 32, ...) sizes

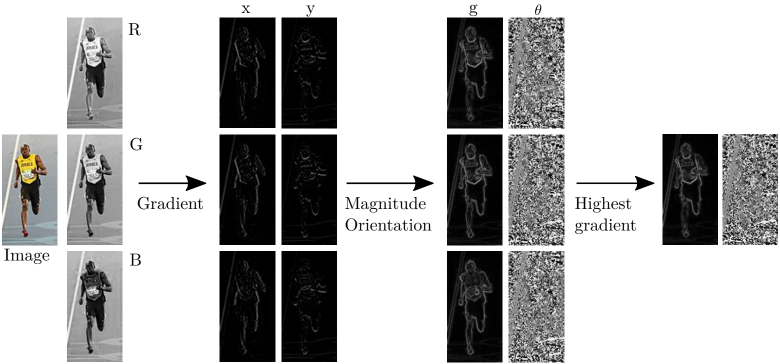


Original Image : 720 x 475

2) Compute gradient images

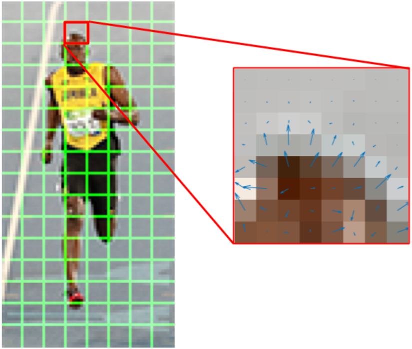
- Gradients obtained by convolution with filters $h_x = [1\ 0\ \text{-}1]$ and $h_y = h_x^T$
- Magnitude of the gradient: $g = \sqrt{gx^2 + gy^2}$

  Orientation of the gradient: $\theta = \arctan(\frac{gy}{gx})$
- In case of a RGB image, the gradient in the highest component is selected



3) Compute histograms of gradients

- The image is divided into cells, generally of size 8x8 pixels
- The orientations are considered regardless of their directions:

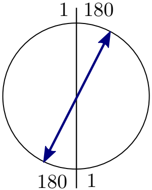


2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

magnitude

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
56	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

orientation



3) Compute histograms of gradients

- 9-bin histogram H built for each cell (a bin corresponds to a $20°$ angle)
- For each θ, the closest bins are proportionnaly filled with magnitude g
- A window of $20°$ is considered so 2 bins $H[i_1]$ and $H[i_2]$ are filled as:

$$i_1 = \mathsf{E}\left(\frac{\theta}{20}\right) \qquad \mathrm{H}[i_1] = \mathrm{H}[i_1] + \frac{20 * (i_1 + 1) - \theta}{20} * g$$

$$i_2 = \mathsf{mod}(i_1 + 1, 9) \qquad \mathrm{H}[i_2] = \mathrm{H}[i_2] + \frac{\theta - 20 * i_1}{20} * g$$

magnitude g

2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

orientation θ

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
56	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

170 0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170

H	0	0	0	0	0	0	0	0	0
	0	1	2	3	4	5	6	7	8

3) Compute histograms of gradients

- 9-bin histogram H built for each cell (a bin corresponds to a 20° angle)
- For each θ, the closest bins are proportionnaly filled with magnitude g
- A window of 20° is considered so 2 bins $H[i_1]$ and $H[i_2]$ are filled as:

$$i_1 = \mathrm{E}\left(\frac{\theta}{20}\right) \qquad \mathrm{H}[i_1] = \mathrm{H}[i_1] + \frac{20*(i_1+1)-\theta}{20} * g$$

$$i_2 = \mathrm{mod}(i_1+1, 9) \qquad \mathrm{H}[i_2] = \mathrm{H}[i_2] + \frac{\theta - 20*i_1}{20} * g$$

magnitude g

2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

orientation θ

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
56	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

170 0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170

H	0	1	2	3	4	5	6	7	8
	0	0	0	0	0	0	0	0	0

3) Compute histograms of gradients

- 9-bin histogram H built for each cell (a bin corresponds to a 20° angle)
- For each θ, the closest bins are proportionnaly filled with magnitude g
- A window of 20° is considered so 2 bins $H[i_1]$ and $H[i_2]$ are filled as:

$$i_1 = \mathsf{E}\left(\frac{\theta}{20}\right) \qquad \mathrm{H}[i_1] = \mathrm{H}[i_1] + \frac{20 * (i_1 + 1) - \theta}{20} * g$$

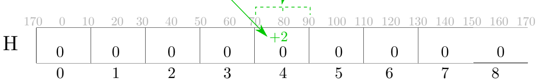
$$i_2 = \mathsf{mod}(i_1 + 1, 9) \qquad \mathrm{H}[i_2] = \mathrm{H}[i_2] + \frac{\theta - 20 * i_1}{20} * g$$

magnitude g

2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

orientation θ

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
56	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110



3) Compute histograms of gradients

- 9-bin histogram H built for each cell (a bin corresponds to a 20° angle)
- For each θ, the closest bins are proportionnaly filled with magnitude g
- A window of 20° is considered so 2 bins $H[i_1]$ and $H[i_2]$ are filled as:

$$i_1 = \mathsf{E}\left(\frac{\theta}{20}\right) \qquad \mathrm{H}[i_1] = \mathrm{H}[i_1] + \frac{20 * (i_1 + 1) - \theta}{20} * g$$

$$i_2 = \mathsf{mod}(i_1 + 1, 9) \qquad \mathrm{H}[i_2] = \mathrm{H}[i_2] + \frac{\theta - 20 * i_1}{20} * g$$

magnitude g

2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

orientation θ

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
56	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

170 0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170

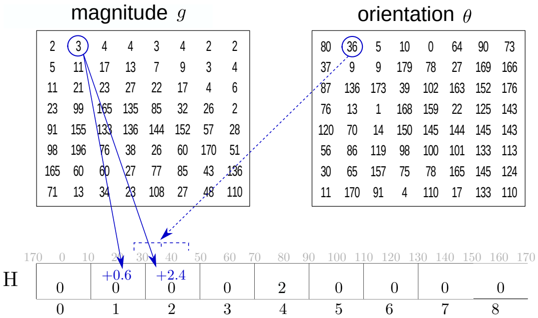
	0	1	2	3	4	5	6	7	8
H	0	0	0	0	2	0	0	0	0

3) Compute histograms of gradients

- 9-bin histogram H built for each cell (a bin corresponds to a 20° angle)
- For each θ, the closest bins are proportionnaly filled with magnitude g
- A window of 20° is considered so 2 bins $H[i_1]$ and $H[i_2]$ are filled as:

$$i_1 = \mathsf{E}\left(\frac{\theta}{20}\right) \qquad \mathrm{H}[i_1] = \mathrm{H}[i_1] + \frac{20 * (i_1 + 1) - \theta}{20} * g$$

$$i_2 = \mathsf{mod}(i_1 + 1, 9) \qquad \mathrm{H}[i_2] = \mathrm{H}[i_2] + \frac{\theta - 20 * i_1}{20} * g$$



3) Compute histograms of gradients

- 9-bin histogram H built for each cell (a bin corresponds to a $20°$ angle)
- For each θ, the closest bins are proportionnaly filled with magnitude g
- A window of $20°$ is considered so 2 bins $H[i_1]$ and $H[i_2]$ are filled as:

$$i_1 = \mathsf{E}\left(\frac{\theta}{20}\right) \qquad \mathrm{H}[i_1] = \mathrm{H}[i_1] + \frac{20 * (i_1 + 1) - \theta}{20} * g$$

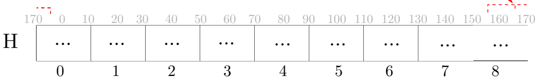
$$i_2 = \mathsf{mod}(i_1 + 1, 9) \qquad \mathrm{H}[i_2] = \mathrm{H}[i_2] + \frac{\theta - 20 * i_1}{20} * g$$

magnitude g

2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

orientation θ

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
56	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110



3) Compute histograms of gradients

- 9-bin histogram H built for each cell (a bin corresponds to a $20°$ angle)
- For each θ, the closest bins are proportionnaly filled with magnitude g
- A window of $20°$ is considered so 2 bins $H[i_1]$ and $H[i_2]$ are filled as:

$$i_1 = \mathsf{E}\left(\frac{\theta}{20}\right) \qquad \mathrm{H}[i_1] = \mathrm{H}[i_1] + \frac{20*(i_1+1)-\theta}{20} * g$$

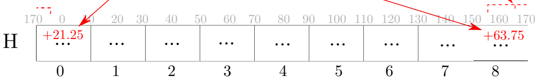
$$i_2 = \mathsf{mod}(i_1+1, 9) \qquad \mathrm{H}[i_2] = \mathrm{H}[i_2] + \frac{\theta - 20*i_1}{20} * g$$

magnitude g

2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

orientation θ

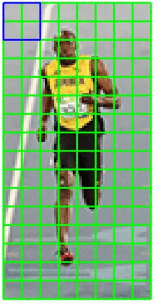
80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
56	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110



4) Block normalization

- Local gradients are sensitive to image illumination $\rightarrow$ normalization
- Cells are gathered into overlapping blocks of 2x2 cells
- In each block the 4 histograms are concatenated $\rightarrow$ 36-bin histogram $\mathrm{H_b}$
- Each value of the 36-bin histogram is normalized by the norm:

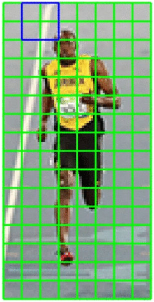
$$\mathrm{H_b}[i] = \frac{\mathrm{H_b}[i]}{\sqrt{\sum_{i=1}^{36} \mathrm{H_b}[i]^2}}$$



4) Block normalization

- Local gradients are sensitive to image illumination → normalization
- Cells are gathered into overlapping blocks of 2x2 cells
- In each block the 4 histograms are concatenated → 36-bin histogram $\mathrm{H_b}$
- Each value of the 36-bin histogram is normalized by the norm:

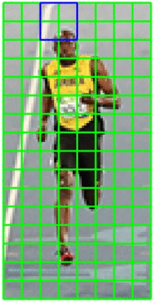
$$\mathrm{H_b}[i] = \frac{\mathrm{H_b}[i]}{\sqrt{\sum_{i=1}^{36} \mathrm{H_b}[i]^2}}$$



4) Block normalization

- Local gradients are sensitive to image illumination → normalization
- Cells are gathered into overlapping blocks of 2x2 cells
- In each block the 4 histograms are concatenated → 36-bin histogram $\mathrm{H_b}$
- Each value of the 36-bin histogram is normalized by the norm:

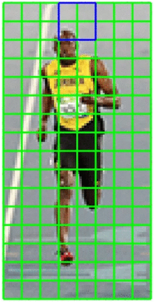
$$\mathrm{H_b}[i] = \frac{\mathrm{H_b}[i]}{\sqrt{\sum_{i=1}^{36} \mathrm{H_b}[i]^2}}$$



4) Block normalization

- Local gradients are sensitive to image illumination → normalization
- Cells are gathered into overlapping blocks of 2x2 cells
- In each block the 4 histograms are concatenated → 36-bin histogram $\mathrm{H_b}$
- Each value of the 36-bin histogram is normalized by the norm:

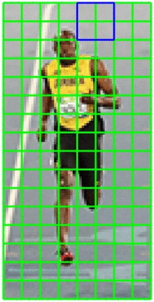
$$\mathrm{H_b}[i] = \frac{\mathrm{H_b}[i]}{\sqrt{\sum_{i=1}^{36} \mathrm{H_b}[i]^2}}$$



4) Block normalization

- Local gradients are sensitive to image illumination → normalization
- Cells are gathered into overlapping blocks of 2x2 cells
- In each block the 4 histograms are concatenated → 36-bin histogram $\mathrm{H_b}$
- Each value of the 36-bin histogram is normalized by the norm:

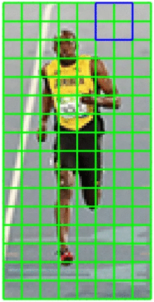
$$\mathrm{H_b}[i] = \frac{\mathrm{H_b}[i]}{\sqrt{\sum_{i=1}^{36} \mathrm{H_b}[i]^2}}$$



4) Block normalization

- Local gradients are sensitive to image illumination → normalization
- Cells are gathered into overlapping blocks of 2x2 cells
- In each block the 4 histograms are concatenated → 36-bin histogram $\mathrm{H_b}$
- Each value of the 36-bin histogram is normalized by the norm:

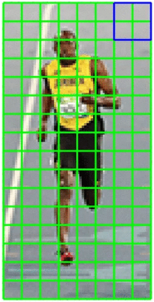
$$\mathrm{H_b}[i] = \frac{\mathrm{H_b}[i]}{\sqrt{\sum_{i=1}^{36} \mathrm{H_b}[i]^2}}$$



4) Block normalization

- Local gradients are sensitive to image illumination → normalization
- Cells are gathered into overlapping blocks of 2x2 cells
- In each block the 4 histograms are concatenated → 36-bin histogram $\mathrm{H_b}$
- Each value of the 36-bin histogram is normalized by the norm:

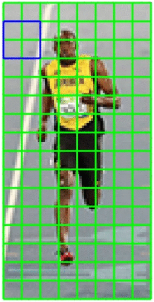
$$\mathrm{H_b}[i] = \frac{\mathrm{H_b}[i]}{\sqrt{\sum_{i=1}^{36} \mathrm{H_b}[i]^2}}$$



4) Block normalization

- Local gradients are sensitive to image illumination $\rightarrow$ normalization
- Cells are gathered into overlapping blocks of 2x2 cells
- In each block the 4 histograms are concatenated $\rightarrow$ 36-bin histogram $\mathrm{H_b}$
- Each value of the 36-bin histogram is normalized by the norm:

$$\mathrm{H_b}[i] = \frac{\mathrm{H_b}[i]}{\sqrt{\sum_{i=1}^{36} \mathrm{H_b}[i]^2}}$$



$\rightarrow$ On this example on size 128x64 pixels:

16x8 cells (8x8 pixels)

15x7 blocks (16x16 pixels)

$\rightarrow$ The whole image information is summarized by:

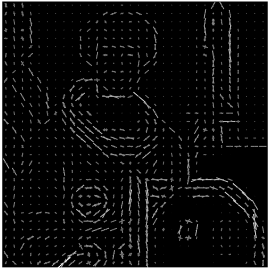
a 15x7x36=3780x1 feature vector

4) Block normalization

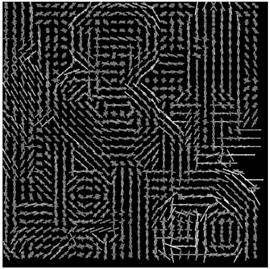
- Local gradients are sensitive to image illumination → normalization
- Cells are gathered into overlapping blocks of 2x2 cells
- In each block the 4 histograms are concatenated → 36-bin histogram $\mathrm{H_b}$
- Each value of the 36-bin histogram is normalized by the norm:

$$\mathrm{H_b}[i] = \frac{\mathrm{H_b}[i]}{\sqrt{\sum_{i=1}^{36} \mathrm{H_b}[i]^2}}$$

HOG cells



HOG blocks



Summary

- HOG is a feature descriptor based on gradient's magnitude and direction
- The local information is gathered in a 9-bin histogram to describe patterns
- Block normalization can be further used to make the model more optimal and less biased by illumination differences → 36-bin histogram.
- However, HOG are not rotation-invariant
- All the histograms form a HOG descriptor that can be used as feature for a recognition or classification system (faces, cars, clothes, actions, etc).

HOG vs (Dense) SIFT

- SIFT descriptor is meant to describe keypoints (4x4 HOG in a centered Gaussian 16x16 window) and is rotation-invariant → efficient for matching
- A dense SIFT (computed for each pixel) would be very costly while non-robust to noise
- HOG is meant to describe patterns, and is computed on the whole image with normalization mechanisms → should perform better for classification

Curse of Dimensionality

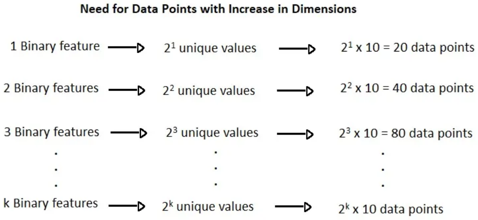
Definition: Problems that arise when working with high-dimensional data

It describes the explosive nature of increasing data dimensions and the exponential increase in computational efforts required for its processing and/or analysis

In our context, it may be correlated to:

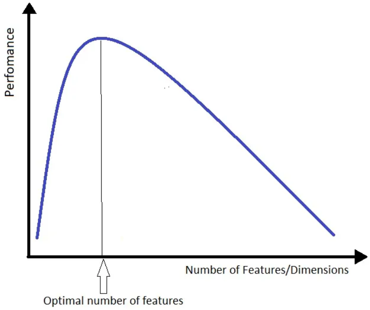
- The number of image to process
- The image dimension
- The number of features, *i.e.* size of the descriptor.

As the dimensionality increases, the number of data points required for good performance of any machine learning algorithm increases exponentially.



Curse of Dimensionality

With a fixed number of training samples, the predictive power of any classifier first increases as the number of dimensions increases, but after a certain value of number of dimensions, the performance deteriorates.



Example of descriptor sizes for a 128x64 pixels image:

- Block-wise LBP: (histogram size x number of cells) = 256 x 16x8 = 32768
- HOG: (histogram size x number of blocks) = 36 x 15x7 = 3780

Curse of dimensionality

Effet on distance functions:

For any point A, lets assume $dist_{\min(A)}$ and $dist_{\max(A)}$ are the respective minimum and maximum distances between A and another point.

In 1D, 2D, or 3D data space:

$$(dist_{\max}(A) - dist_{\min}(A))/dist_{\min}(A) > 0$$

But, as the number of dimensions increases:

$$\lim_{dim \to \infty} (dist_{\max}(A) - dist_{\min}(A))/dist_{\min}(A) \to 0$$

$$\text{so} \qquad dist_{\max}(A) \approx dist_{\min}(A)$$

$\rightarrow$ Classification algorithms based on the distance measure, including k-NN (k-Nearest Neighbor) tend to fail when the number of dimensions is very high.

Solution: Reduce the feature space dimension by:

- Manually selecting the most useful subset of features
- Using dimension reduction methods such as PCA

Principal Component Analysis

Exploratory multivariate analysis method, introduced by Hotelling in 1933 following ideas from Pearson (1901)

Application domains: Data Science, Physics, Biology, Sociology, Marketing, Quality Control, ...

Data: set of individuals characterized by a set of quantitative variables

Objective: to summarize the initial variables using a small number of synthetic variables (the *principal components*) obtained from linear combinations

Use:

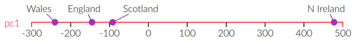
- Evaluate the similarities between individuals
- Condense the representation of data while preserving as much as possible their global organization
- Allow a visualization of the preponderant organization of data thanks to a projection on a low-dimensional space (2D, 3D)
- Prepare other analyzes by eliminating redundant variables and the directions in which the variance of the data is very small

Principal Component Analysis

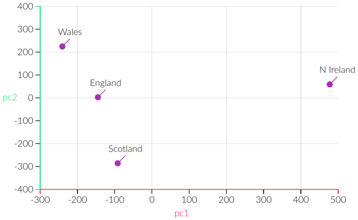
Example 1: Eating in the UK

Data with 17 variables... How to visualize similarity between countries?

	England	N Ireland	Scotland	Wales
Alcoholic drinks	375	135	458	475
Beverages	57	47	53	73
Carcase meat	245	267	242	227
Cereals	1472	1494	1462	1582
Cheese	105	66	103	103
Confectionery	54	41	62	64
Fats and oils	193	209	184	235
Fish	147	93	122	160
Fresh fruit	1102	674	957	1137
Fresh potatoes	720	1033	566	874
Fresh Veg	253	143	171	265
Other meat	685	586	750	803
Other Veg	488	355	418	570
Processed potatoes	198	187	220	203
Processed Veg	360	334	337	365
Soft drinks	1374	1506	1572	1256
Sugars	156	139	147	175



1D projection



2D projection

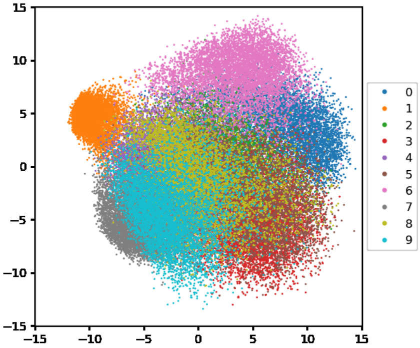
Source: https://setosa.io/ev/principal-component-analysis/

Principal Component Analysis

Example 2: Digit Images

Directly consider the image pixels as variables

Example of digit images



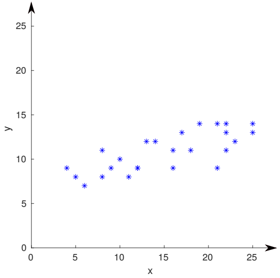
2D projection

Source: https://www.researchgate.net/publication/365623663_
Confidence_is_key_Using_Gaussian_Process_Classifiers_to_improv

On a 2D example

Data $\mathbf{X}$, containing n samples/individuals, described by $p = 2$ variables

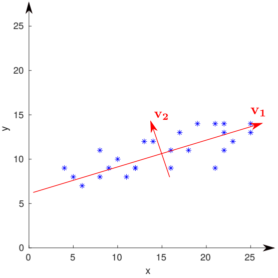
$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \dots \\ \mathbf{x}_n^T \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ \dots & \dots \\ x_{n,1} & x_{n,2} \end{bmatrix} = \begin{bmatrix} 6 & 7 \\ 11 & 8 \\ 16 & 9 \\ 22 & 11 \\ \dots & \dots \\ 12 & 9 \\ 8 & 8 \\ 5 & 8 \end{bmatrix}$$



On a 2D example

Data $\mathbf{X}$, containing n samples/individuals, described by $p = 2$ variables

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \dots \\ \mathbf{x}_n^T \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ \dots & \dots \\ x_{n,1} & x_{n,2} \end{bmatrix} = \begin{bmatrix} 6 & 7 \\ 11 & 8 \\ 16 & 9 \\ 22 & 11 \\ \dots & \dots \\ 12 & 9 \\ 8 & 8 \\ 5 & 8 \end{bmatrix}$$

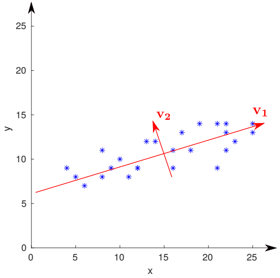


How to find a discriminant projection base $\{\mathbf{v}_1, \mathbf{v}_2\}$?

On a 2D example

Data $\mathbf{X}$, containing n samples/individuals, described by $p = 2$ variables

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \dots \\ \mathbf{x}_n^T \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ \dots & \dots \\ x_{n,1} & x_{n,2} \end{bmatrix} = \begin{bmatrix} 6 & 7 \\ 11 & 8 \\ 16 & 9 \\ 22 & 11 \\ \dots & \dots \\ 12 & 9 \\ 8 & 8 \\ 5 & 8 \end{bmatrix}$$



How to find a discriminant projection base $\{\mathbf{v}_1, \mathbf{v}_2\}$?

Criteria: Maximizing the dispersion/inertia, *i.e.*, the distance after projection

Solution: The eigen vectors of the covariance matrix

Why the eigen vectors of the covariance matrix ?

Objective: Find the subspaces maximizing the dispersion/inertia of the points

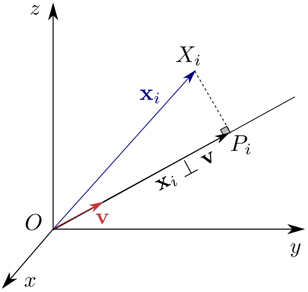
On a line: Which line maximizes the inertia of the orthogonal projections?

- Line given by the unitary vector $\mathbf{v}$ such as $\|\mathbf{v}\|_2 = \mathbf{v}^T\mathbf{v} = 1$
- Projections on line $\mathbf{v}$

$$d(O, P_i) = \mathbf{x}_i^T\mathbf{v} = \|\mathbf{x}_i\|_2 \cos(\mathbf{x}_i, \mathbf{v})$$

- Maximizing the dispersion of projections on the line is similar to optimizing the adjustment of the point cloud since

$$d(O, P_i)^2 = d(O, X_i)^2 - d(X_i, P_i)^2$$



- Total inertia of the points projected on the line:

$$\sum_{i=1}^{n} d^2(O, P_i) = \sum_{i=1}^{n} \left(\mathbf{x}_i^T\mathbf{v}\right)^2$$

Why the eigen vectors of the covariance matrix ?

Objective: Maximize $\sum_{i=1}^{n} \left(\mathbf{x}_i^T \mathbf{v}\right)^2$ with $\mathbf{v}^T\mathbf{v} = 1$

$$\sum_{i=1}^{n} \left(\mathbf{x}_i^T \mathbf{v}\right)^2 = \sum_{i=1}^{n} \left(\mathbf{x}_i^T \mathbf{v}\right)^T \left(\mathbf{x}_i^T \mathbf{v}\right) = \sum_{i=1}^{n} \mathbf{v}^T \left(\mathbf{x}_i \mathbf{x}_i^T\right) \mathbf{v} = \mathbf{v}^T \left[\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^T\right] \mathbf{v}$$

$$= \mathbf{v}^T \mathbf{X}^T \mathbf{X} \mathbf{v}$$

Resolution: Lagrangien multiplier method under the constraint $\mathbf{v}^T\mathbf{v} = 1$

- Lagrangien: $L(\mathbf{v}) = \mathbf{v}^T \mathbf{X}^T \mathbf{X} \mathbf{v} - \alpha(\mathbf{v}^T\mathbf{v} - 1)$
- Partial derivative: $\frac{\partial L}{\partial \mathbf{v}}(\mathbf{v}) = 2\mathbf{X}^T \mathbf{X} \mathbf{v} - 2\alpha \mathbf{v}$
- Solving $\frac{\partial L}{\partial \mathbf{v}}(\mathbf{v}) = 0$ to get extremas: $\frac{\partial L}{\partial \mathbf{v}}(\mathbf{v}) = 0 \Leftrightarrow \mathbf{X}^T \mathbf{X} \mathbf{v} = \alpha \mathbf{v}$

  $\rightarrow$ Standard eigen vector/value problem ($\alpha = \lambda$)

  $\rightarrow \mathbf{v}^T \mathbf{X}^T \mathbf{X} \mathbf{v} = \lambda \mathbf{v}^T \mathbf{v} = \lambda$ so λ is the highest eigen value and $\mathbf{v}$ the corresponding eigen vector

- The exact same resolution gives all the other vectors of the projection base maximizing the inertia, as the eigen vectors of $\mathbf{X}^T\mathbf{X}$
- In practice, eigen vectors of the covariance matrix $\mathbf{C} = \frac{\mathbf{X}^T\mathbf{X}}{n-1}$

Principle

Data $\mathbf{X}$, containing n samples/individuals, described by p variables

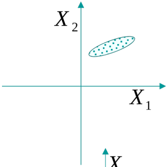
$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \dots \\ \mathbf{x}_n \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,p} \\ x_{2,1} & x_{2,2} & \dots & x_{2,p} \\ \dots & \dots & \dots & \dots \\ x_{n,1} & x_{n,2} & \dots & x_{n,p} \end{bmatrix}$$

with variable average $\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_{i,j}$ and variance $\sigma_j^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_{i,j}-\mu_j)^2$

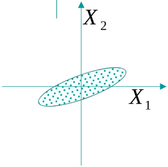
Steps to perform a PCA:

1) Data standardization: centering (μ_j=0), normalization (σ_j=1)
2) Computation of the covariance matrix $\mathbf{C} = \frac{\mathbf{X}^T\mathbf{X}}{n-1}$
3) Computation of a discriminant projection base (eigen vectors/values of $\mathbf{C}$)
4) Data projection towards a potentially lower space ($k << p$)

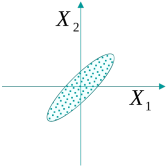
1) Data standardization



- **General PCA:** Raw input data
  - Analysis based on the natural zero of some variables



- **Centered PCA:** Centered variables ($\mu_j = 0$)
  - Easier interpretation based on the gravity center
  - With variables that are directly comparable
  - A high variance variable may largely impact the PCA



- **Standardized PCA:** Centered ($\mu_j = 0$) and reduced ($\sigma_j = 1$) variables
  - All variables are normalized and directly comparable
  - A noise variable gets the same variance as informative ones

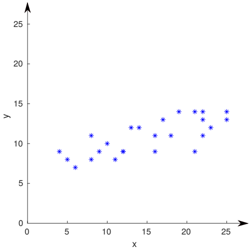
Conclusion

$\rightarrow$ Depends on the type of data
$\rightarrow$ Generally easier to center the data
$\rightarrow$ If the variables are in different units, the reduction seems necessary

1) Data standardization: on a 2D example

Raw input

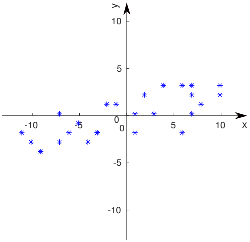
$$\mathbf{X}$$



$$\mathbf{X} = \begin{bmatrix} 6 & 7 \\ 11 & 8 \\ 16 & 9 \\ 22 & 11 \\ \dots & \dots \\ 12 & 9 \\ 8 & 8 \\ 5 & 8 \end{bmatrix} \quad \begin{array}{l} \mu_1 = 15.2 \\ \mu_2 = 10.8 \\ \sigma_1 = 6.37 \\ \sigma_2 = 2.17 \end{array}$$

Centered

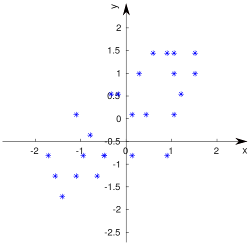
$$\mathbf{X}_{:,j} \leftarrow \mathbf{X}_{:,j} - \mu_j$$



$$\mathbf{X} = \begin{bmatrix} -9.2 & -3.8 \\ -4.2 & -2.8 \\ 0.8 & -1.8 \\ 6.8 & 0.2 \\ \dots & \dots \\ -3.2 & -1.8 \\ -7.2 & -2.8 \\ -10.2 & -2.8 \end{bmatrix} \quad \begin{array}{l} \mu_1 = 0 \\ \mu_2 = 0 \\ \sigma_1 = 6.4 \\ \sigma_2 = 2.2 \end{array}$$

Centered and reduced

$$\mathbf{X}_{:,j} \leftarrow \frac{\mathbf{X}_{:,j} - \mu_j}{\sigma_j}$$



$$\mathbf{X} = \begin{bmatrix} -1.41 & -1.71 \\ -0.64 & -1.26 \\ 0.13 & -0.81 \\ 1.05 & 0.09 \\ \dots & \dots \\ -0.49 & -0.81 \\ -1.10 & -1.26 \\ -1.56 & -1.26 \end{bmatrix} \quad \begin{array}{l} \mu_1 = 0 \\ \mu_2 = 0 \\ \sigma_1 = 1 \\ \sigma_2 = 1 \end{array}$$

2) Computation of the covariance matrix

Covariance: For two vectors $\mathbf{x}$ and $\mathbf{y}$ of size n of mean μ_x and μ_y

$$\text{Cov}(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n} (x_i - \mu_x)(y_i - \mu_y)}{n-1}$$

$\rightarrow$ Measures the correlation between the two vectors

Interpretation:

- $|\text{Cov}(\mathbf{x}, \mathbf{y})| >> 0$: indicates high correlation between variables
- $\text{Cov}(\mathbf{x}, \mathbf{y}) > 0$: the variables vary in the same manner (positive correlation)
- $\text{Cov}(\mathbf{x}, \mathbf{y}) < 0$: the variables vary in opposite manner (negative correlation)
- $\text{Cov}(\mathbf{x}, \mathbf{y}) = 0$: the variables are independent

Remarks:

- $\text{Cov}(\mathbf{x}, \mathbf{x}) = \sigma(\mathbf{x})^2$ the variance of $\mathbf{x}$
- $\text{Cov}(\mathbf{x}, \mathbf{y}) = \text{Cov}(\mathbf{y}, \mathbf{x})$

2) Computation of the covariance matrix

Covariance matrix:

$$\mathbf{C} = \begin{bmatrix} \mathrm{Cov}(\mathbf{x},\mathbf{x}) & \mathrm{Cov}(\mathbf{x},\mathbf{y}) \\ \mathrm{Cov}(\mathbf{y},\mathbf{x}) & \mathrm{Cov}(\mathbf{y},\mathbf{y}) \end{bmatrix} = \begin{bmatrix} \sigma(\mathbf{x})^2 & \mathrm{Cov}(\mathbf{x},\mathbf{y}) \\ \mathrm{Cov}(\mathbf{y},\mathbf{x}) & \sigma(\mathbf{y})^2 \end{bmatrix}$$

$\rightarrow$ $\mathbf{C}$ is symmetric and summarizes the variance information

In our context:

Data $\mathbf{X}_{n \times p}$, containing n individuals, described by p variables

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,p} \\ x_{2,1} & x_{2,2} & \dots & x_{2,p} \\ \dots & \dots & \dots & \dots \\ x_{n,1} & x_{n,2} & \dots & x_{n,p} \end{bmatrix}$$

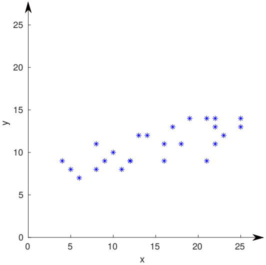
Covariance matrix $\mathbf{C}_{p \times p}$ computed on all variables $\mathbf{x}_{:,\mathbf{j}}$:

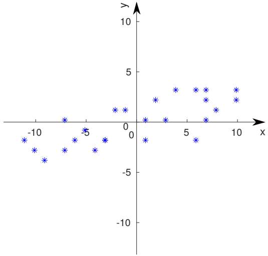
$$\mathbf{C} = \frac{1}{n-1}\mathbf{X}^T\mathbf{X}$$

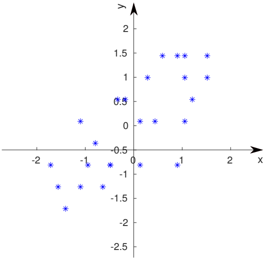
$\rightarrow$ If the data $\mathbf{X}$ has been centered ($\mathbf{x}_{:,\mathbf{j}} - \mu_j$), we exactly get the previous covariance definition

2) Computation of the covariance matrix: on a 2D example

Raw input	Centered	Centered and reduced
$\mathbf{X}$	$\mathbf{X}_{:,j} \leftarrow \mathbf{X}_{:,j} - \mu_j$	$\mathbf{X}_{:,j} \leftarrow \frac{\mathbf{X}_{:,j} - \mu_j}{\sigma_j}$







$$\mathbf{C} = \begin{bmatrix} 281.6 & 181.4 \\ 181.4 & 126.4 \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} 42.2 & 10.9 \\ 10.9 & 4.92 \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} 1 & 0.7542 \\ 0.7542 & 1 \end{bmatrix}$$

3) Computation of a discriminant projection base

$\mathbf{C}$ being built from $\mathbf{X}^T\mathbf{X}$, it is positive semidefinite.
→ It admits a matrix decomposition using eigen values and vectors such that:

$$\mathbf{C} = \mathbf{P\Lambda P}^{-1} = \mathbf{P\Lambda P}^T$$

where:

- $\mathbf{P}$ contains the eigen vectors $\{\mathbf{v_i}\}$ that form an orthonormal basis

$$\mathbf{P} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_p] \quad \text{with} \quad \mathbf{v}_i^T\mathbf{v}_j = 0 \text{ if } i \neq j, \text{ and } \|\mathbf{v}_i\|_2 = 1$$

- $\mathbf{\Lambda}$ contains the eigen values $\{\lambda_i\}$

$$\mathbf{\Lambda} = \text{Diag}(\{\lambda_i\}) = \begin{bmatrix} \lambda_1 & 0 & ... & 0 \\ 0 & \lambda_2 & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & \lambda_p \end{bmatrix} \quad \text{with} \quad \lambda_1 \geq \lambda_2 \geq ... \geq \lambda_p \geq 0$$

and $\omega_{\lambda_i} = \frac{\lambda_i}{\sum_{j=1}^{p} \lambda_j}$ the weight of each eigen value/vector

→ Can be computed using `eigs` or `svd` (for singular value decompositions)

3) Computation of a discriminant projection base: on a 2D example

On centered data $\mathbf{X}$

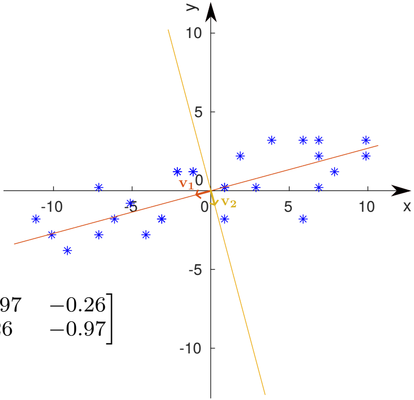
$$\mathbf{C} = \begin{bmatrix} 42.2 & 10.9 \\ 10.9 & 4.92 \end{bmatrix}$$

$\rightarrow$ eigen values and vectors

$$\mathbf{C} = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^T$$

$$= \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \end{bmatrix}$$

$$= \begin{bmatrix} -0.97 & 0.26 \\ -0.26 & 0.97 \end{bmatrix} \begin{bmatrix} 45.16 & 0 \\ 0 & 1.98 \end{bmatrix} \begin{bmatrix} -0.97 & -0.26 \\ 0.26 & -0.97 \end{bmatrix}$$



$$\lambda_1 = 45.16 \qquad \mathbf{v}_1 = [-0.97, -0.26]^T \qquad \omega_{\lambda_1} = 0.96$$
$$\lambda_2 = 1.98 \qquad \mathbf{v}_2 = [0.26, -0.97]^T \qquad \omega_{\lambda_2} = 0.04$$

4) Data projection

Eigen vector matrix: $\mathbf{P} = [\mathbf{v}_1 \ \mathbf{v}_2 \ ... \ \mathbf{v}_k \ ... \ \mathbf{v}_p]$

Projection matrix : $\mathbf{P}' = [\mathbf{v}_1 \ \mathbf{v}_2 \ ... \ \mathbf{v}_k \ \mathbf{0} \ ... \ \mathbf{0}]$

$\rightarrow$ To keep only the first k components, with the highest signification.
$\rightarrow$ Each component contains information from correlated variables.

Projection to the new space:

For an individual $\mathbf{x}$: $\mathbf{x}' = \mathbf{P}'^T \mathbf{x}$

For the whole data $\mathbf{X}$: $\mathbf{X}'^T = \mathbf{P}'^T \mathbf{X}^T$

$\rightarrow$ For a new individual $\mathbf{x} \notin \mathbf{X}$, possibility to directly apply the projection (after eventual standardization), without recomputing the projection matrix $\mathbf{P}$.

Reprojection to the original space:

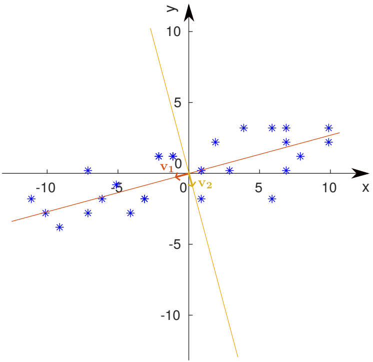
For an individual $\mathbf{x}$: $\mathbf{x}'' = \mathbf{P}\mathbf{x}' = \mathbf{P}\mathbf{P}'^T \mathbf{x}$

For the whole data $\mathbf{X}$: $\mathbf{X}''^T = \mathbf{P}\mathbf{X}'^T = \mathbf{P}\mathbf{P}'^T \mathbf{X}^T$

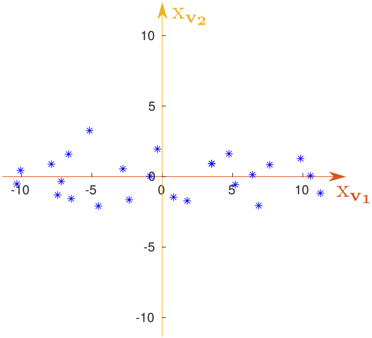
4) Data projection: on a 2D example

So for each individual $\mathbf{x}$, using a 2 dimension projection space

$$\mathbf{x}' = \mathbf{P}'^T \mathbf{x} = \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \end{bmatrix} \mathbf{x} \quad \Leftrightarrow \quad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} -0.97 & -0.26 \\ 0.26 & -0.97 \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$



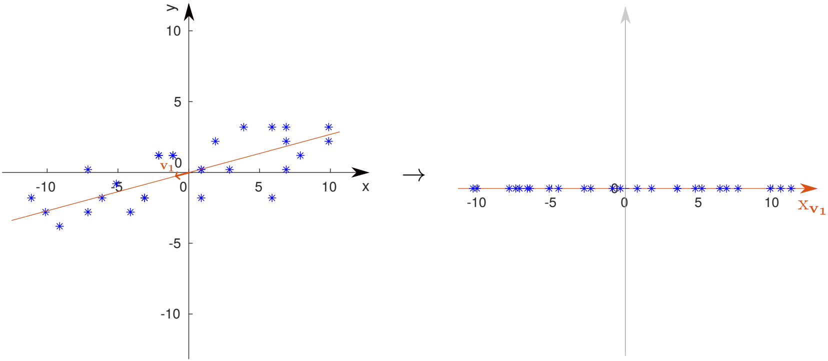
$\rightarrow$



4) Data projection: on a 2D example

So for each individual $\mathbf{x}$, using a 1 dimension projection space

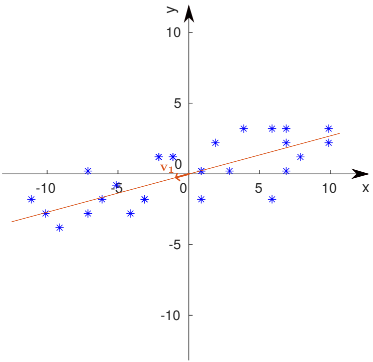
$$\mathbf{x}' = \mathbf{P}'^T \mathbf{x} = \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{0} \end{bmatrix} \mathbf{x} \quad \Leftrightarrow \quad \begin{pmatrix} x' \\ 0 \end{pmatrix} = \begin{bmatrix} -0.97 & -0.26 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$



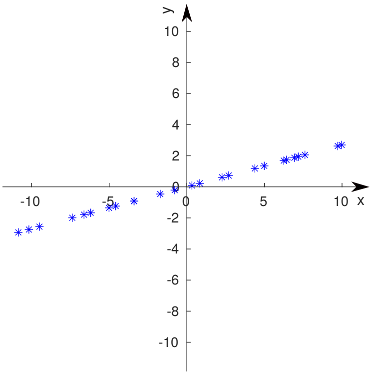
4) Data projection: on a 2D example

Reprojection after simplification (1 dimensional space)

$$\mathbf{x}'' = \mathbf{P}\mathbf{P}'^T\mathbf{x} = [\mathbf{v}_1 \ \mathbf{v}_2] \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{0} \end{bmatrix} \mathbf{x} \ \Leftrightarrow \ x'' = \begin{bmatrix} -0.97 & 0.26 \\ -0.26 & 0.97 \end{bmatrix} \begin{bmatrix} -0.97 & -0.26 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$



$\rightarrow$



A practical example: Superheros

The variables:

1) Have super powers (0 or 1)
2) Wear tights (between 1 and 3)
3) Work in teams (between 1 and 10)
4) Have a specific gear (between 1 and 10)
5) Man / Woman (1 or 0)

A practical example: Superheros

The individuals:

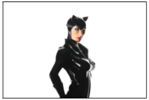
Superman

Batman

Spiderman

Hulk

Ironman



Catwoman

X-or

Daredevil

Wonderwoman

Bioman

X-men

Tortues ninjas

A practical example: Superheros

The data:

	1 (powers)	**2** (tights)	**3** (team)	**4** (gear)	**5** (man/woman)
Superman	1	3	2	2	1
Batman	0	3	7	10	1
Spiderman	1	3	2	2	1
Hulk	1	1	1	1	1
Ironman	0	1	3	10	1
Catwoman	0	3	2	3	0
X-or	0	1	2	10	1
Daredevil	0	3	2	3	1
Wonderwoman	1	2	3	9	0
Bioman	0	3	10	10	0.6
X-men	1	2	8	7	0.5
Tortues Ninja	0	1	10	7	0.8

A practical example: Superheros

The data:

	1 (powers)	**2** (tights)	**3** (team)	**4** (gear)	**5** (man/woman)
Superman	1	3	2	2	1
Batman	0	3	7	10	1
Spiderman	1	3	2	2	1
Hulk	1	1	1	1	1
Ironman	0	1	3	10	1
Catwoman	0	3	2	3	0
X-or	0	1	2	10	1
Daredevil	0	3	2	3	1
Wonderwoman	1	2	3	9	0
Bioman	0	3	10	10	0.6
X-men	1	2	8	7	0.5
Tortues Ninja	0	1	10	7	0.8
mean	0.41	2.17	4.33	6.17	0.74

A practical example: Superheros

The data: after centering $(\mathbf{X}_{:,j} = \mathbf{X}_{:,j} - \mu_j)$

	1 (powers)	**2** (tights)	**3** (team)	**4** (gear)	**5** (man/woman)
Superman	0.58	0.83	-2.33	-4.17	0.26
Batman	-0.42	0.83	2.67	3.83	0.26
Spiderman	0.58	0.83	-2.33	-4.17	0.26
Hulk	0.58	-1.17	-3.33	-5.17	0.26
Ironman	-0.42	-1.17	-1.33	3.83	0.26
Catwoman	-0.42	0.83	-2.33	-3.17	-0.74
X-or	-0.42	-1.17	-2.33	3.83	0.26
Daredevil	-0.42	0.83	-2.33	-3.17	0.26
Wonderwoman	0.58	-0.17	-1.33	2.83	-0.74
Bioman	-0.42	0.83	5.67	3.83	-0.14
X-men	0.58	-0.17	3.67	0.83	-0.24
Tortues Ninja	-0.42	-1.17	5.67	0.83	0.058
mean	0	0	0	0	0

A practical example: Superheros

The data: after centering $(\mathbf{X}_{:,j} = \mathbf{X}_{:,j} - \mu_j)$

	1 (powers)	2 (tights)	3 (team)	4 (gear)	5 (man/woman)
Superman	0.58	0.83	-2.33	-4.17	0.26
Batman	-0.42	0.83	2.67	3.83	0.26
Spiderman	0.58	0.83	-2.33	-4.17	0.26
Hulk	0.58	-1.17	-3.33	-5.17	0.26
Ironman	-0.42	-1.17	-1.33	3.83	0.26
Catwoman	-0.42	0.83	-2.33	-3.17	-0.74
X-or	-0.42	-1.17	-2.33	3.83	0.26
Daredevil	-0.42	0.83	-2.33	-3.17	0.26
Wonderwoman	0.58	-0.17	-1.33	2.83	-0.74
Bioman	-0.42	0.83	5.67	3.83	-0.14
X-men	0.58	-0.17	3.67	0.83	-0.24
Tortues Ninja	-0.42	-1.17	5.67	0.83	0.058
mean	0	0	0	0	0
variance	0.49	0.83	2.9	3.3	0.31

A practical example: Superheros

The data: after centering and normalization $\left(\mathbf{X}_{:,j} = \frac{\mathbf{x}_{:,j} - \mu_j}{\sigma_j}\right)$

	1 (powers)	**2** (tights)	**3** (team)	**4** (gear)	**5** (man/woman)
Superman	1.2	1	-0.79	-1.3	0.83
Batman	-0.86	1	0.91	1.2	0.83
Spiderman	1.2	1	-0.79	-1.3	0.83
Hulk	1.2	-1.4	-1.1	-1.6	0.83
Ironman	-0.86	-1.4	-0.45	1.2	0.83
Catwoman	-0.86	1	-0.79	-0.96	-2.4
X-or	-0.86	-1.4	-0.79	1.2	0.83
Daredevil	-0.86	1	-0.79	-0.96	0.83
Wonderwoman	1.2	-0.2	-0.45	0.86	-0.2.4
Bioman	-0.86	1	1.9	1.2	-0.46
X-men	1.2	-0.2	1.2	0.25	-0.78
Tortues Ninja	-0.86	-1.14	1.9	0.25	0.19
mean	0	0	0	0	0
variance	1	1	1	1	1

A practical example: Superheros

The covariance matrix:

$$\mathbf{C} = \begin{bmatrix} 1.00 & 0.03 & -0.29 & -0.47 & -0.09 \\ 0.03 & 1.00 & 0.01 & -0.27 & -0.17 \\ -0.29 & 0.01 & 1.00 & 0.53 & -0.11 \\ -0.47 & -0.27 & 0.53 & 1.00 & -0.11 \\ -0.10 & -0.17 & -0.11 & -0.11 & 1.00 \end{bmatrix}$$

Eigen values and vectors: $(\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_5 \geq 0)$

$$\begin{array}{lll} \lambda_1 = 1.91 & \mathbf{v}_1 = [0.51, 0.19, -0.54, -0.64, 0.06]^T & \omega_{\lambda_1} = 0.38 \\ \lambda_2 = 1.20 & \mathbf{v}_2 = [-0.09, -0.64, -0.25, 0.01, 0.72]^T & \omega_{\lambda_2} = 0.24 \\ \lambda_3 = 0.91 & \mathbf{v}_3 = [0.50, -0.65, -0.04, 0.18, -0.54]^T & \omega_{\lambda_3} = 0.18 \\ \lambda_4 = 0.64 & \mathbf{v}_4 = [0.60, 0.08, 0.69, -0.05, 0.39]^T & \omega_{\lambda_4} = 0.13 \\ \lambda_5 = 0.32 & \mathbf{v}_5 = [0.35, 0.34, -0.41, 0.74, 0.20]^T & \omega_{\lambda_5} = 0.07 \end{array}$$

A practical example: Superheros

Eigen vector matrix:

$$\mathbf{P} = \begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 & \mathbf{v}_4 & \mathbf{v}_5\end{bmatrix} = \begin{bmatrix} 0.51 & -0.09 & 0.50 & 0.60 & 0.35 \\ 0.19 & -0.64 & -0.65 & 0.08 & 0.34 \\ -0.54 & -0.25 & -0.04 & 0.69 & -0.41 \\ -0.64 & 0.01 & 0.18 & -0.05 & 0.74 \\ 0.06 & 0.72 & -0.54 & 0.39 & 0.20 \end{bmatrix}$$

Projection matrix ($p = 5 \rightarrow k = 3$) :

$$\mathbf{P}' = \begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 & \mathbf{0} & \mathbf{0}\end{bmatrix} = \begin{bmatrix} 0.51 & -0.09 & 0.50 & 0 & 0 \\ 0.19 & -0.64 & -0.65 & 0 & 0 \\ -0.54 & -0.25 & -0.04 & 0 & 0 \\ -0.64 & 0.01 & 0.18 & 0 & 0 \\ 0.06 & 0.72 & -0.54 & 0 & 0 \end{bmatrix}$$

$\rightarrow$ To keep only the first $k = 3$ components, with the highest signification.
$\rightarrow$ Each component contains information from correlated variables.

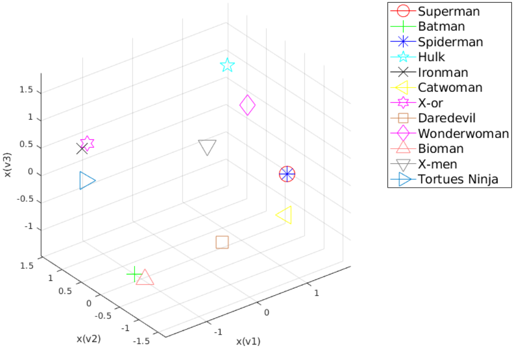
Projection to the new space:

For an individual x: $\quad x' = \mathbf{P}'^T \bar{x}$

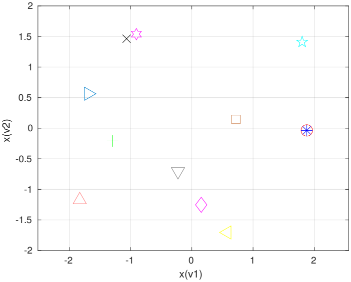
For the whole data X: $\quad \mathbf{X}'^T = \mathbf{P}'^T \bar{\mathbf{X}}^T$

A practical example: Superheros

Visualization:



3D visualization ($k = 3$)



2D visualization ($k = 2$)

→ In 2D or 3D, direct visualization of the proximity between individuals.

→ For a new individual possibility to directly project in the same space.

Summary

- **PCA:** Principal Component Analysis
  Practical tool to measure and visualize the correlation between variables
  Enables to reduce the data dimension (curse of dimensionality)
- **Method:**
  1) Data standardization: depends on the data nature
  2) Covariance matrix: holds the correlation information
  3) Projection base that fits the data (eigen vectors)
  4) Projection into a reduced space
- **Remark:**
  This method does not consider class information (unsupervised)
  Other unsupervised data reduction approaches exist such as:
  - t-SNE (van der Maaten et al. 2008)
  - UMAP (McInnes et al. 2018) that can also be supervised

Practical n°2

- **Data:** Natural textures from [Cimpoi et al., 2014]

  10 different textures

  For each texture, 120 images of size 256x256

- **Histogram of Oriented Gradients (HOG)**
  - Compute the HOG descriptor (`skimage.feature.hog`).
- **Local Binary Pattern (LBP)**
  - Implement the Local Binary Pattern method for $R = 1$, $P = 8$
  - Compute the global histogram (`np.histogram`)
  - Compute block-wise histograms (`np.histogram`)

Practical n°2

- **Supervised classification**
  - The train and test data are stored in 4 files: *I_train.npy*, *Y_train.npy*, *I_test.npy*, *Y_test.npy*.
    For each of the 10 classes, we have 96 train and 24 test images.
    - I_train.npy contains an array of 960 images (256x256x3x960)
    - Y_train.npy contains a class array of size 960x1
    - I_test.npy contains an array of 240 images (256x256x3x240)
    - Y_test.npy contains a class array of size 240x1
  - Compute the desired features for each image to create X_train and X_test of size 960xp and 240xp (with p the size of the descriptor)
  - Evaluate the performance of the descriptor by using a supervised classifier (SVM)
- **Dimension reduction**
  - Apply PCA on the descriptor
  - Does it help to obtain better classification?
  - Measure the computation time for each case

References

Slides inspired from:

- Marc Donias: https://donias.vvv.enseirb-matmeca.fr/ts326.html
- Michaël Clément: https://www.labri.fr/perso/mclement/
- Florent Grélard: https://fgrelard.github.io/#teaching
- Michel Crucianu: http://cedric.cnam.fr/~crucianm/rfmn.html
- Vincent Nozick: https://igm.univ-mlv.fr/~vnozick/teaching/slides/imac2_math/10_pca.pdf
- Serena Yeung: https://ai.stanford.edu/~syyeung/cvweb/tutorials.html
- Victor Powell: https://setosa.io/ev/principal-component-analysis/
- Mrinal Tyagi: https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f
- Shashmi Karanam: https://towardsdatascience.com/curse-of-dimensionality-a-curse-to-machine-learning-c122ee33bfeb
- Wikipédia: https://fr.wikipedia.org/wiki/Motif_binaire_local
- Datahacker: https://datahacker.rs/opencv-circle-detection-hough-transform/
- Niebles & Krishna: http://vision.stanford.edu/teaching/cs131_fall1718/files/14_BoW_bayes.pdf
- Hugo Larochelle: https://info.usherbrooke.ca/hlarochelle/cours/ift603_H2015/description.html