

# MLRF Lecture 01

J. Chazalon, LRE/EPITA, 2025

# Global Image Descriptors

Lecture 01 part 05

# The need for image descriptors

# Issues with methods based on pixel comparison

What is important? What do they consider? **Raw pixels!**

⇒ We want to be able to make use of **domain knowledge!**

*Like sensitivity to shape, or dominant color information.*

They are terribly **slow** and works **only for small images.**

⇒ We want to **summarize an image** to a much smaller vector.

They are **sensible to rotation, scaling**, and many other perturbations.

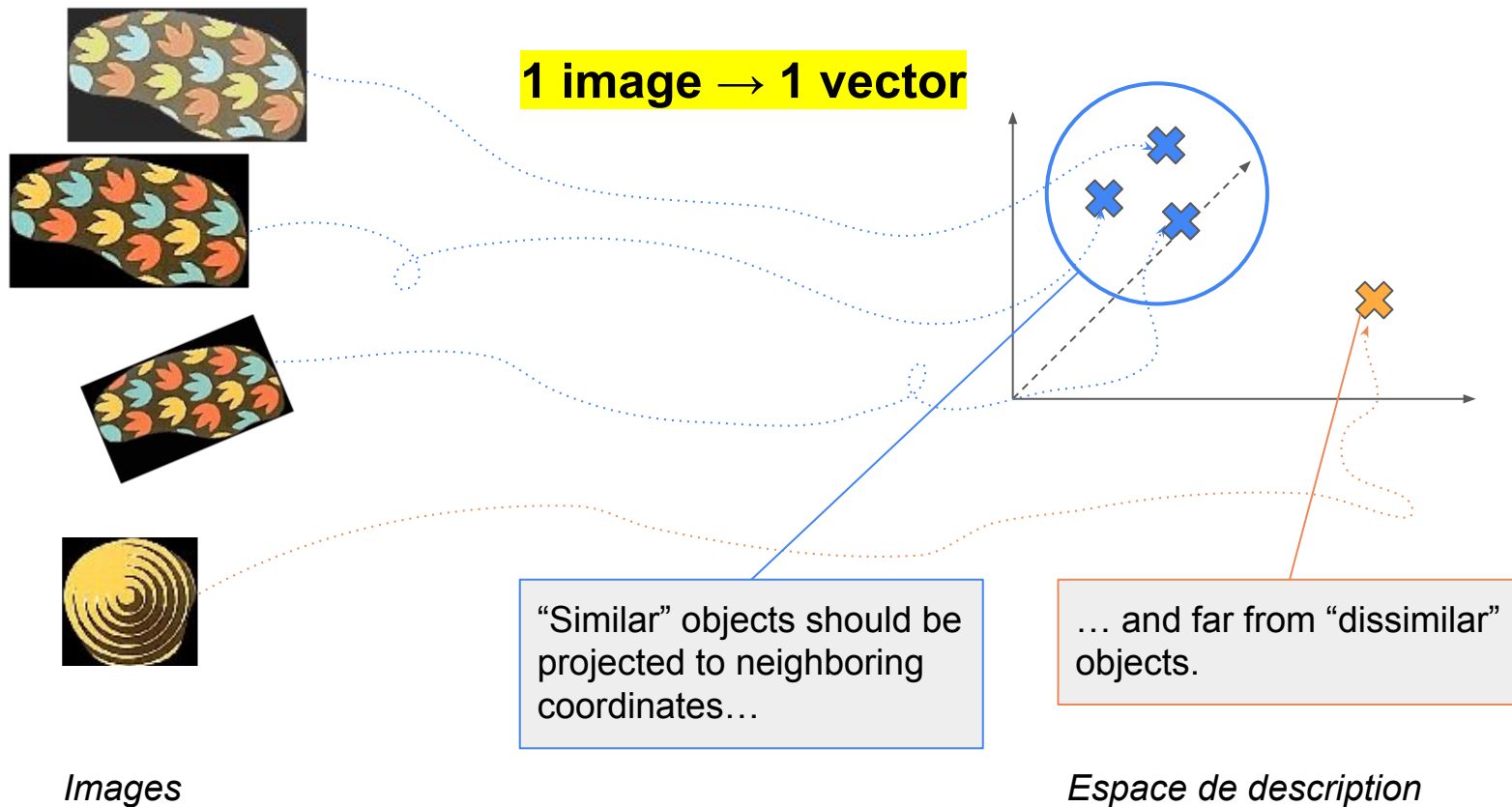
⇒ We want to adjust sensitivity/invariance to perturbations.

*Do we tolerate translation? Rotation? Intensity shift?*

How can we compare different pairs of images? **Metric issues.**

⇒ We want to be able to achieve **more than 1 vs all comparisons.**

# Global descriptors in 1 picture



# Two approaches

## Lecture 4

### Global image descriptors

- Compute **statistics about the content** of the image
- Produce a **single global vector**

**Very attractive because they are very fast to compute and match, but... (see end of section)**

### Bag of Features techniques

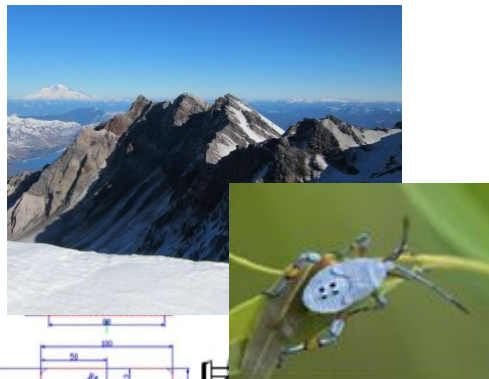
- **Select regions** of interest in the image (may be a variable quantity)
- **Compute descriptors** for each region
- **Index each part** separately (like a text search engine which indexes words)

*It is always possible to build a single descriptor from local descriptors!*

*This technique is the one used in modern image search engines.*

# Image descriptors: Overview

Different sizes and contents  $\Rightarrow$  Different kind of descriptors

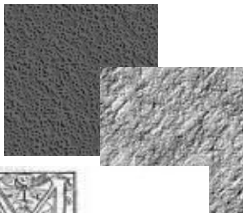


Large images, many parts

$\approx 500 \times 500$  px and more



Complex  
small images



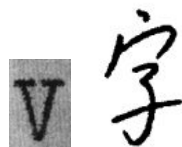
Textured  
areas



Logos



Words



Isolated  
symbols and  
letters



Local image patches,  
lines, etc.

$\approx 20 \times 20$  px and less

# Image descriptors: Overview

**Different sizes and contents  $\Rightarrow$  Different kind of descriptors**

**Different problems  $\Rightarrow$  Different choices**

- Computation / memory constraints
- Which perturbations to we have to tolerate?  
*rotation, translation...*
- What is the expected output?  
*classification, detection, ranking, segmentation...*

**Many, many approaches  $\Rightarrow$  Impossible to list them all**

- Examples of several categories
- Focus on very useful or instructive ones



# Color Histograms

## Color histograms – a very simple global descriptor (of pixels statistics)

High invariance to many transformation

*rotation, scaling thanks to normalization, perspective...*

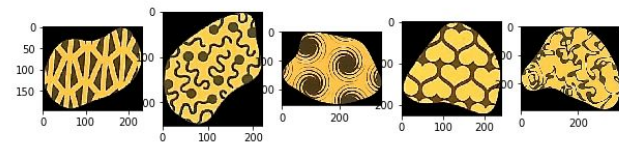
But limited discriminative power

### Easy to implement

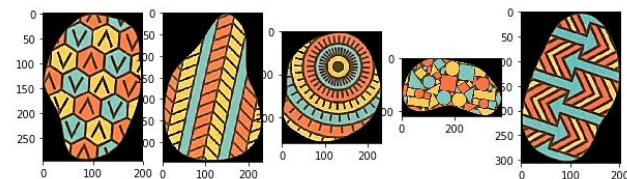
1. Reduce the colors (opt. when performing backprojection)
2. Compute a reduced color histogram on each image
3. Use a distribution distance to compare the descriptors

# Color histograms: Some results on *Twin it!*

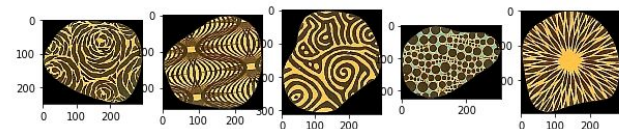
22 86:0.005 257:0.007 156:0.008 13:0.009



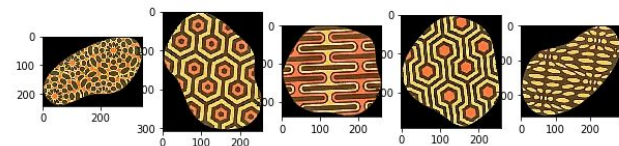
23 378:0.012 320:0.019 297:0.037 263:0.040



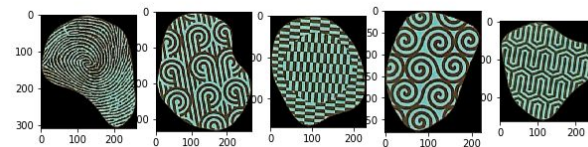
24 331:0.024 302:0.035 323:0.035 271:0.042



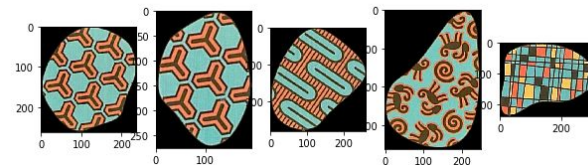
25 376:0.038 197:0.043 66:0.046 205:0.056



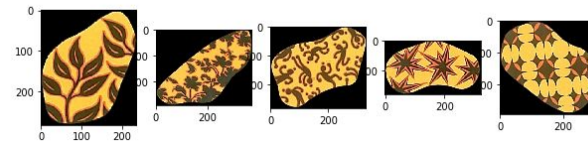
34 318:0.020 42:0.028 242:0.041 102:0.042



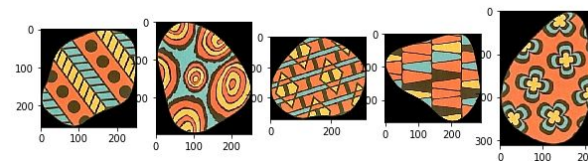
35 219:0.002 335:0.005 362:0.024 251:0.034



36 69:0.025 155:0.028 304:0.030 212:0.034



37 308:0.003 108:0.011 365:0.031 351:0.034



**Timing comparison (1 CPU)**

**Template matching**

Match each pair of image:  
3 hours

**Color Histogram**

Color reduction:  
3 seconds

Compute color histogram for  
all bubbles:  
30 seconds

Compute distance between  
each pair of descriptors:  
2 seconds

# Color histograms: Step by step

## 1: Color reduction

1. Use K-Means or any other clustering technique to find  $N$  useful colors.
2. Project each pixel value on the value of the closest cluster center.

Swain & Ballard 1991

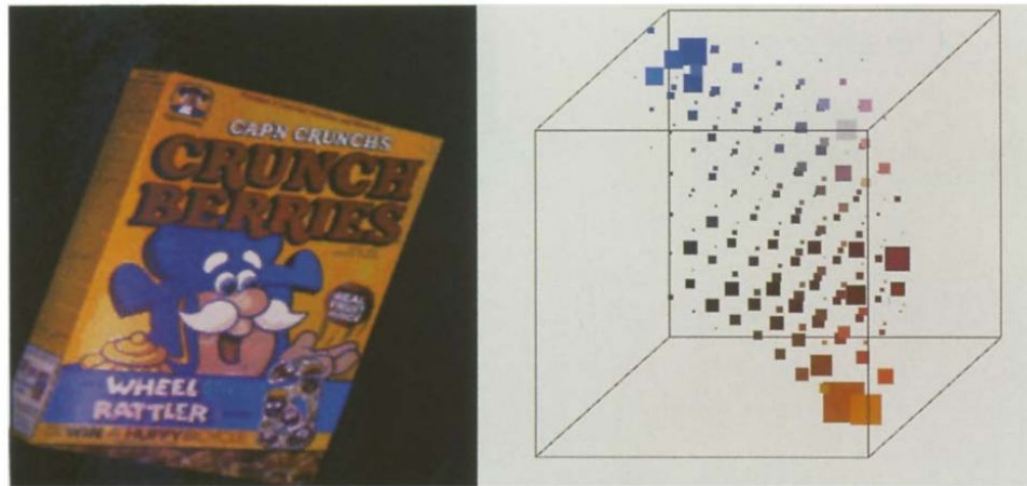


Fig. 1. Left: Image of a Crunchberries cereal box. Right: Three dimensional color histogram of the Crunchberries image with the black background substrated.





↑ 16M colors



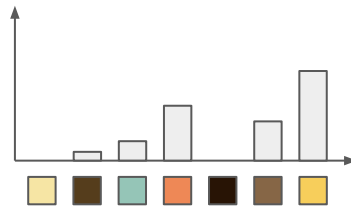
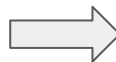
↑ 7 colors (+ white bg)

One possible result on the *Twin it!* poster

# Color histograms: Step by step

## 2: Histogram computation

You already know it.  
(*Normalize it.*)



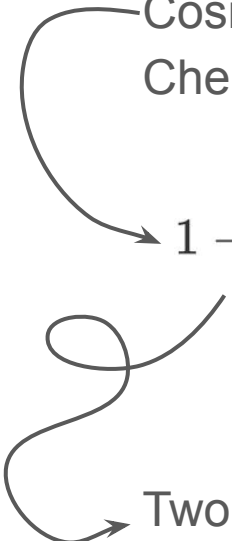


# Color histograms: Step by step

## 3: Descriptor comparison

Many distribution metrics.

Cosine, Euclidean,  
Chebyshev...


$$1 - \frac{u \cdot v}{\|u\|_2 \|v\|_2}.$$

Read, try, compare, learn!

Two histograms =  
Two 1-D vectors!

```
from scipy.spatial.distance import ...
```

Distance functions between two numeric vectors `u` and `v`. Computing distances over a large collection of vectors is inefficient for these functions. Use `pdist` for this purpose.

<code>braycurtis(u, v[, w])</code>	Compute the Bray-Curtis distance between two 1-D arrays.
<code>canberra(u, v[, w])</code>	Compute the Canberra distance between two 1-D arrays.
<code>chebyshev(u, v[, w])</code>	Compute the Chebyshev distance.
<code>cityblock(u, v[, w])</code>	Compute the City Block (Manhattan) distance.
<code>correlation(u, v[, w, centered])</code>	Compute the correlation distance between two 1-D arrays.
<code>cosine(u, v[, w])</code>	Compute the Cosine distance between 1-D arrays.
<code>euclidean(u, v[, w])</code>	Computes the Euclidean distance between two 1-D arrays.
<code>jensenshannon(p, q[, base])</code>	Compute the Jensen-Shannon distance (metric) between two 1-D probability arrays.
<code>mahalanobis(u, v, VI)</code>	Compute the Mahalanobis distance between two 1-D arrays.
<code>minkowski(u, v[, p, w])</code>	Compute the Minkowski distance between two 1-D arrays.
<code>seuclidean(u, v[, w])</code>	Return the standardized Euclidean distance between two 1-D arrays.
<code>squeuclidean(u, v[, w])</code>	Compute the squared Euclidean distance between two 1-D arrays.
<code>wminkowski(u, v[, p, w])</code>	Compute the weighted Minkowski distance between two 1-D arrays.

# Discussion

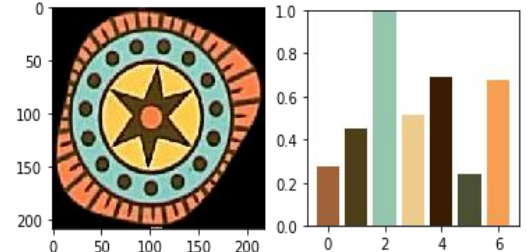
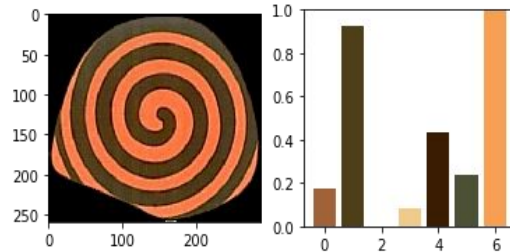
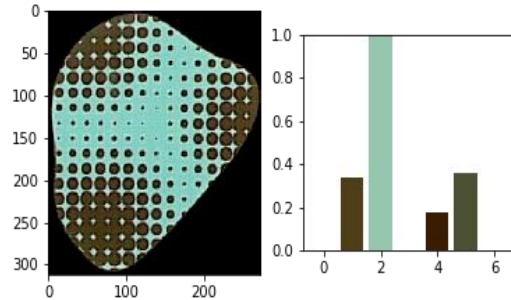
Can you think of other global descriptors we could have implemented for the *Twin it!* case?



# Next practice session

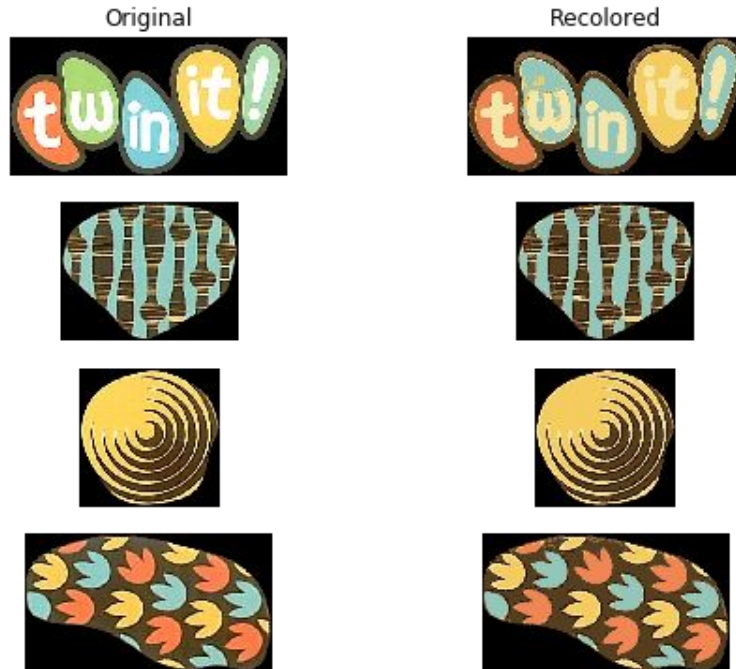
*Twin it!*, again, with a slightly more elaborated approach:

## 1. Pre-select bubbles based on their colors $\Rightarrow$ Color histograms



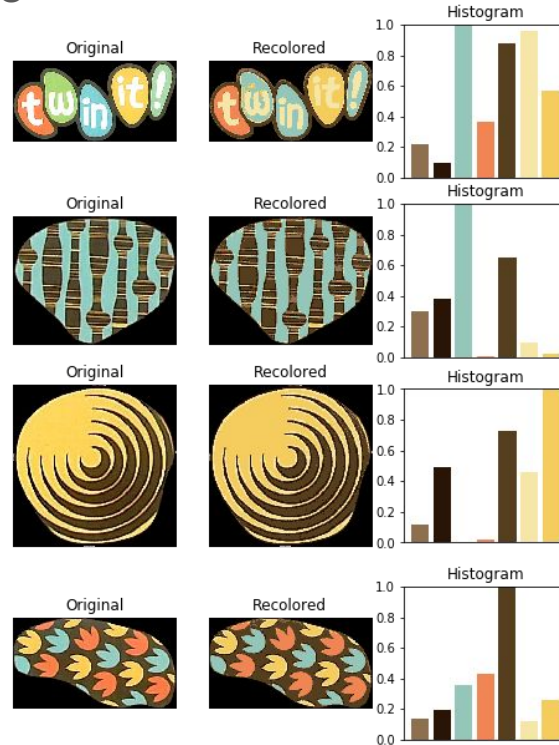
# Color histogram: In details

1.1. Color quantization: reduce the colors of the bubbles.



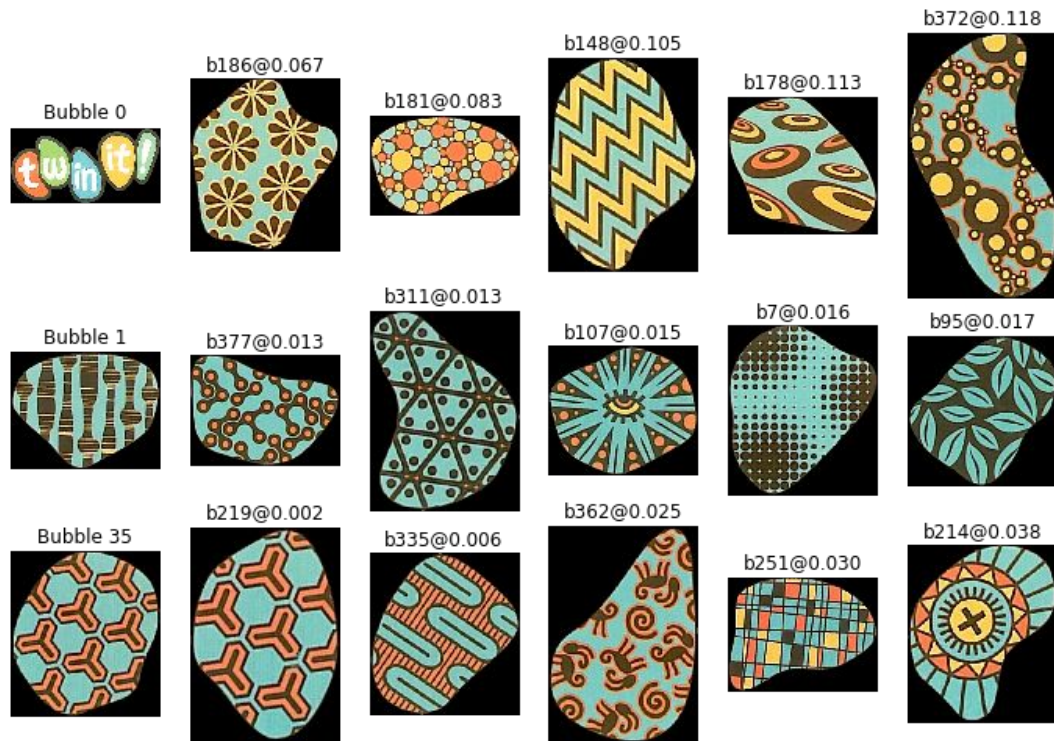
# Color histogram: In details

## 1.2. Compute the color histogram of each bubble.



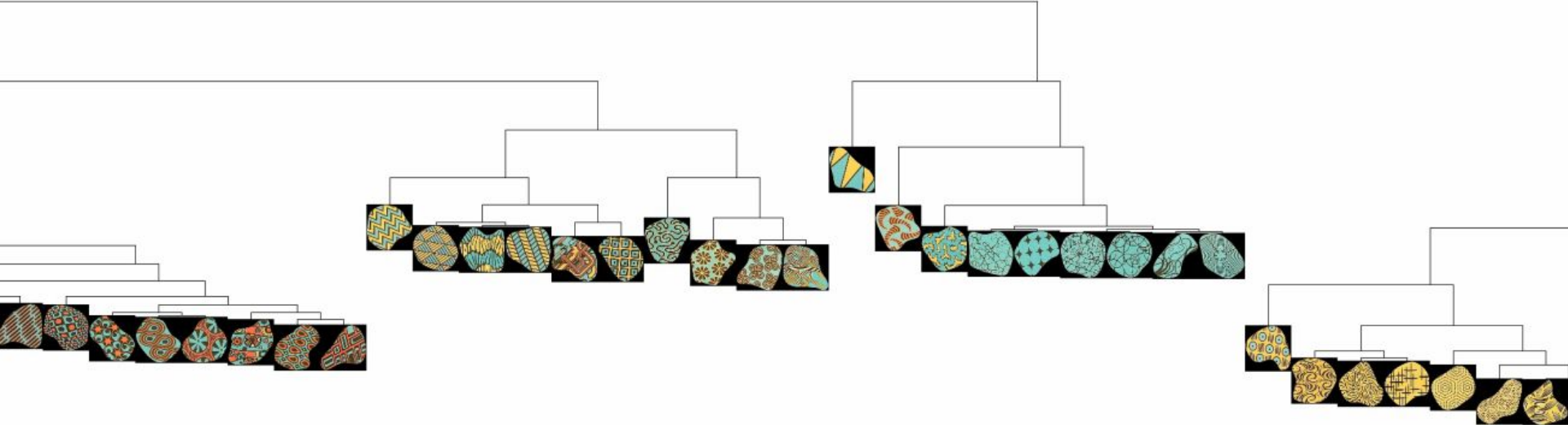
# Color histogram: In details

1.3. Compute the distance matrix between each bubble, using its color histogram.



# Color histogram: In details

1.4. Visualize the bubbles in an interesting way using hierarchical clustering.



# Other global image descriptors

# More global descriptors

## GIST of a scene:

- Oliva, Torralba, “Modeling the shape of the scene: a holistic representation of the spatial envelope”, IJCV’01.
- Douze, Jegou, Sandhawalia, Amsaleg, Schmid, “Evaluation of GIST descriptors for web-scale image search”, CIVR’09.

## CENTRIST: CENsus Transform hISTogram

- Wu, Rehg, “CENTRIST: a visual descriptor for scene categorization”, TPAMI’11.

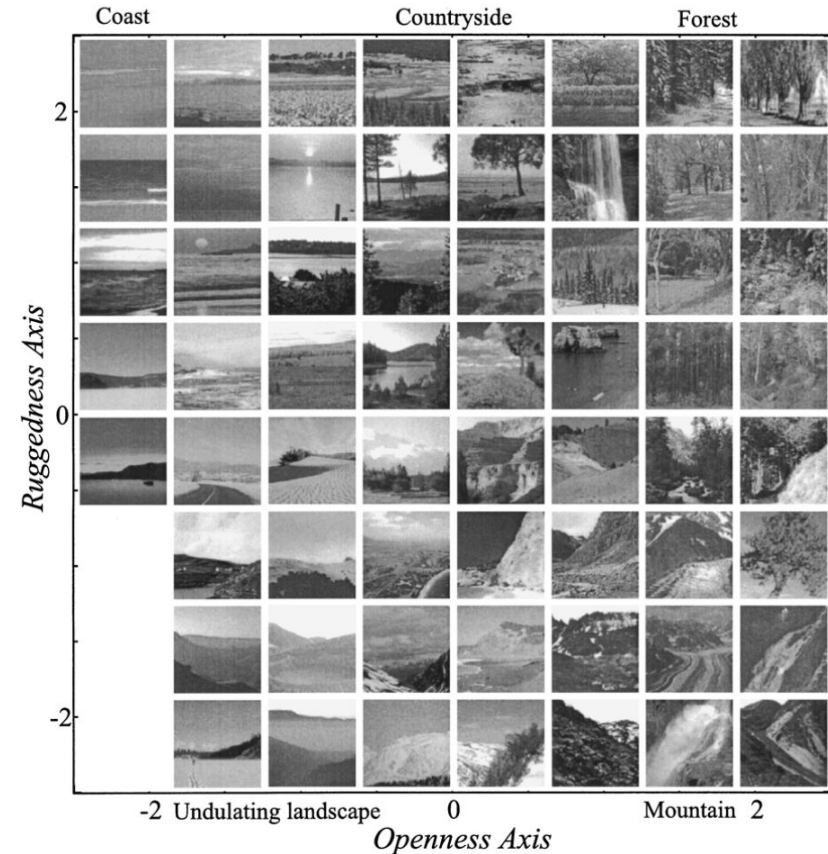


Figure 15. Organization of natural scenes according to the openness and ruggedness properties estimated by the WDSs.

# Global descriptors: drawback

*According to F. Perronnin:*

Highly efficient to compute and to match

⇒ **perfect in theory**

But **robustness vs informativeness tradeoff is hard to set**

(personal conclusion):

- Approaches based on **global image descriptors** are confined to **near-duplicate detection** applications until now.
- Modern search engines use local representations and leverage them.