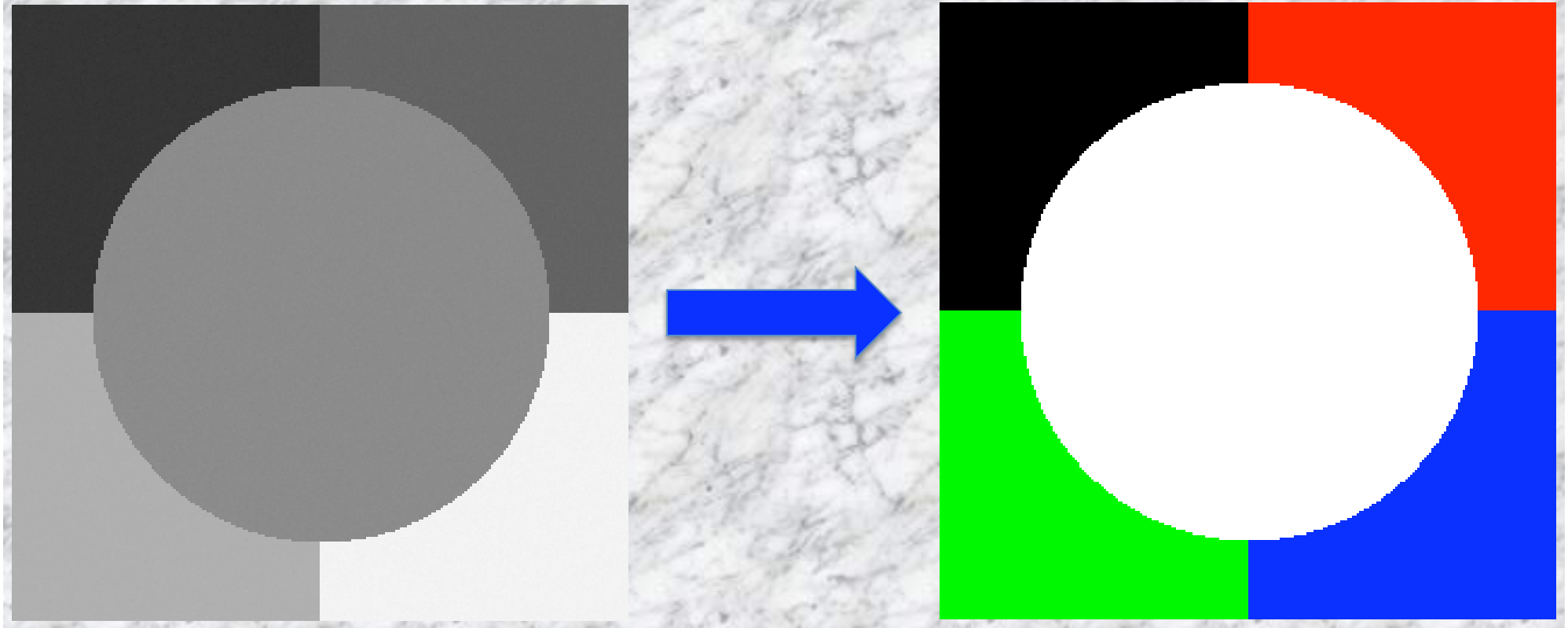
Texture Analysis

Xavier Descombes

Problem of classification



Maximum Likelihood

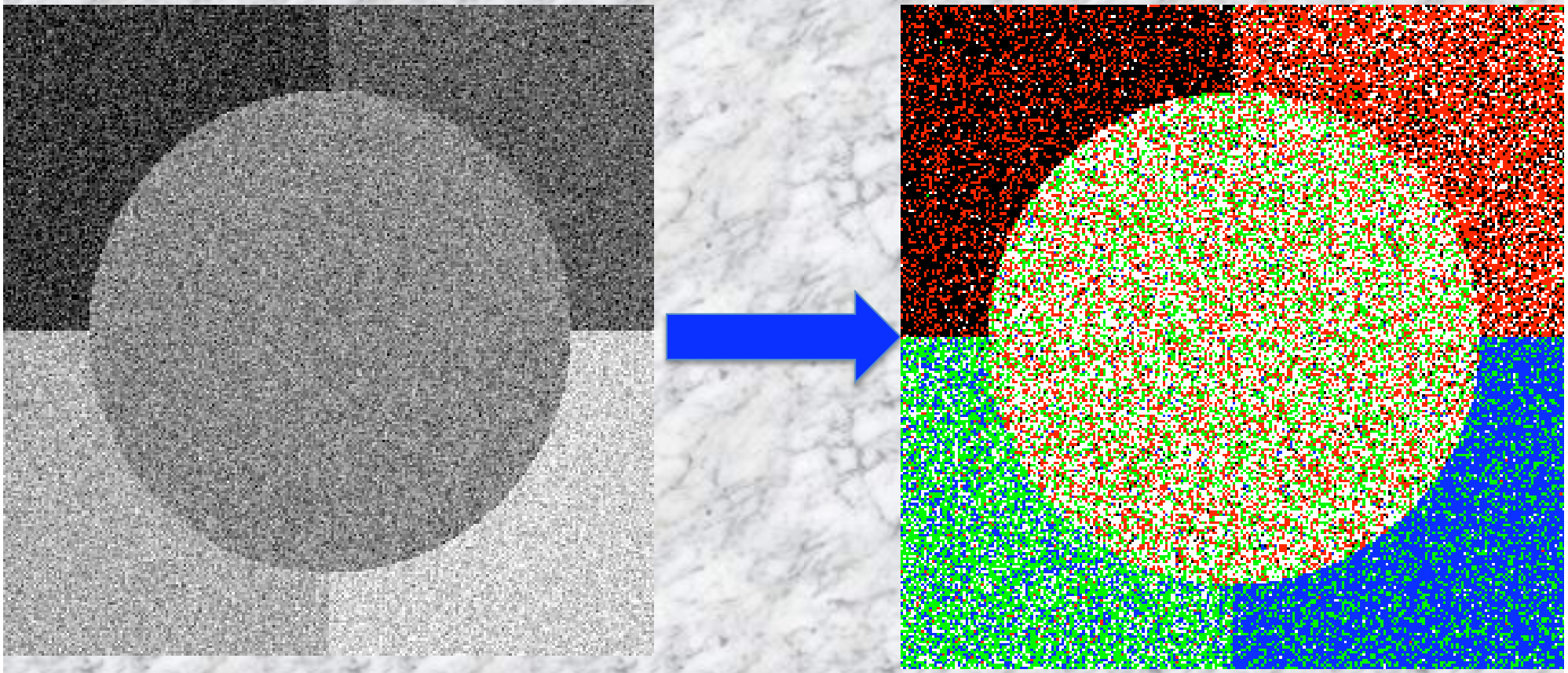
- Define the probability density of classes :

$$\forall c \in C, \forall i \in I, p_c(i) = p(x_s = i \mid y_s = c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} \exp\left[-\frac{(i-\mu_c)^2}{2\sigma_c^2}\right]$$

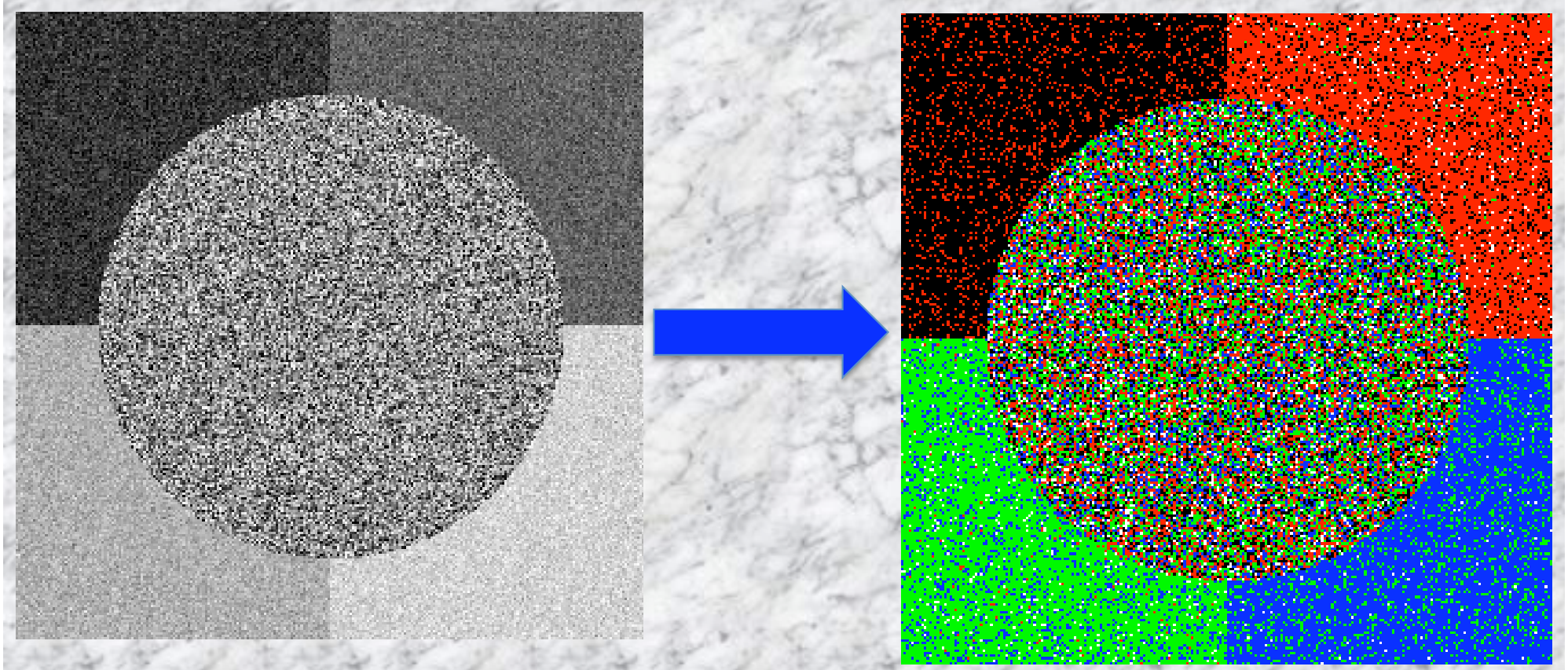
- Maximize this probability:

$$y_s = \arg\max_{c \in C} p_c(i)$$

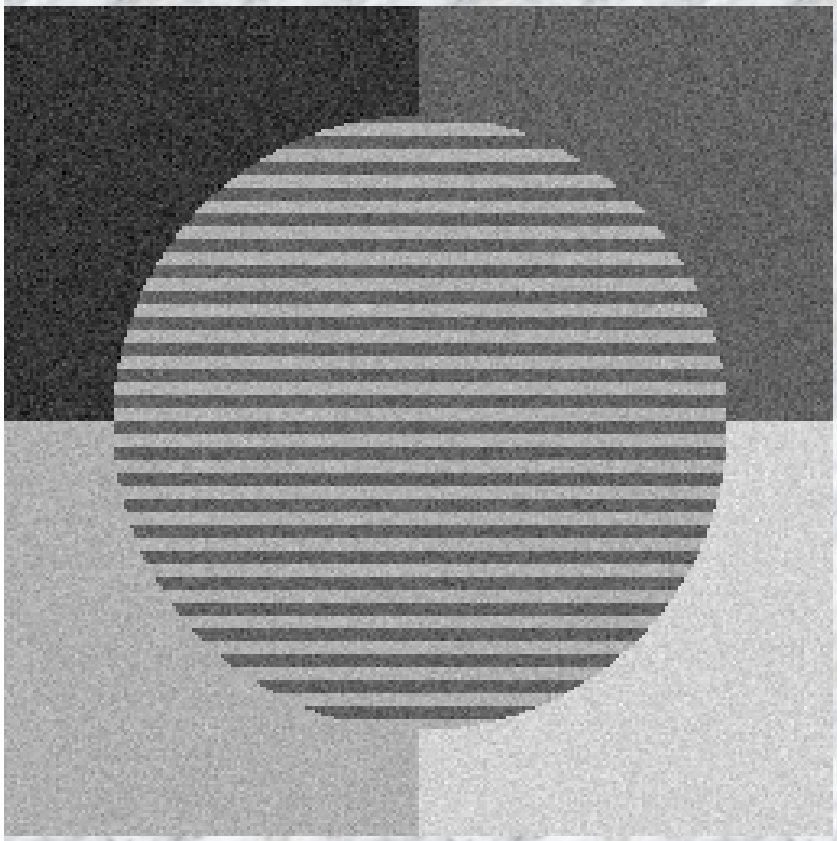
Problem of classification

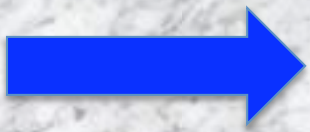


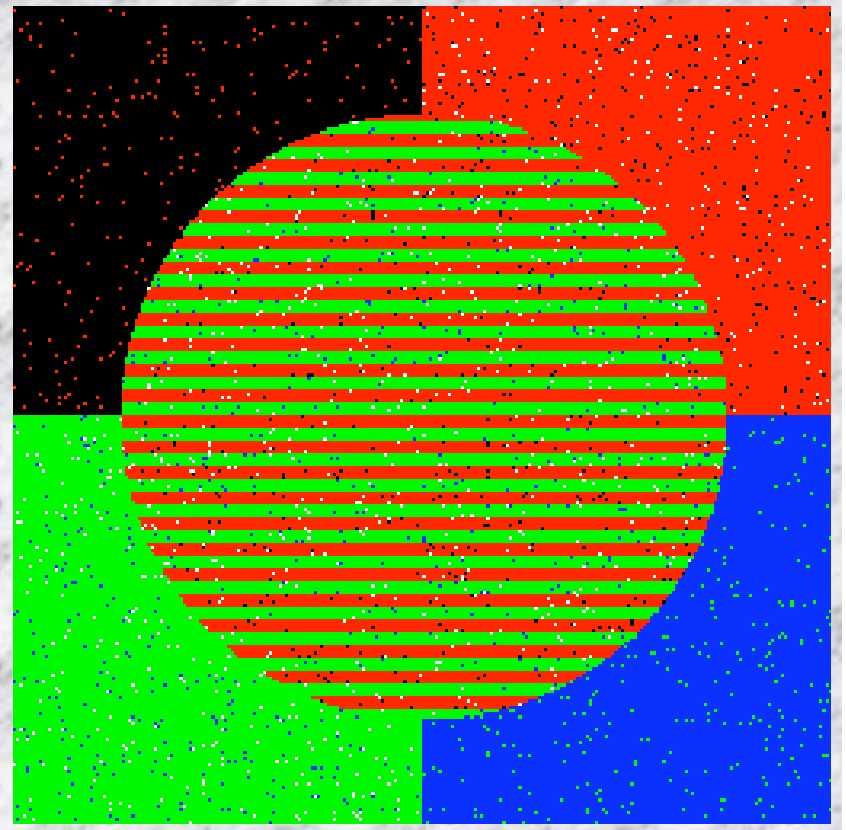
Problem of classification



Problem of classification







Problem of classification : urban area detection

Texture definition

- No exact definition : intuition

- Two main properties :
  - Variability at the pixel scale : inhomogeneity
  - At a coarser scale : homogeneity
  - Depends on the object but also on the image resolution

Analogy with the human visual system :

non uniform area perceived as an homogeneous region

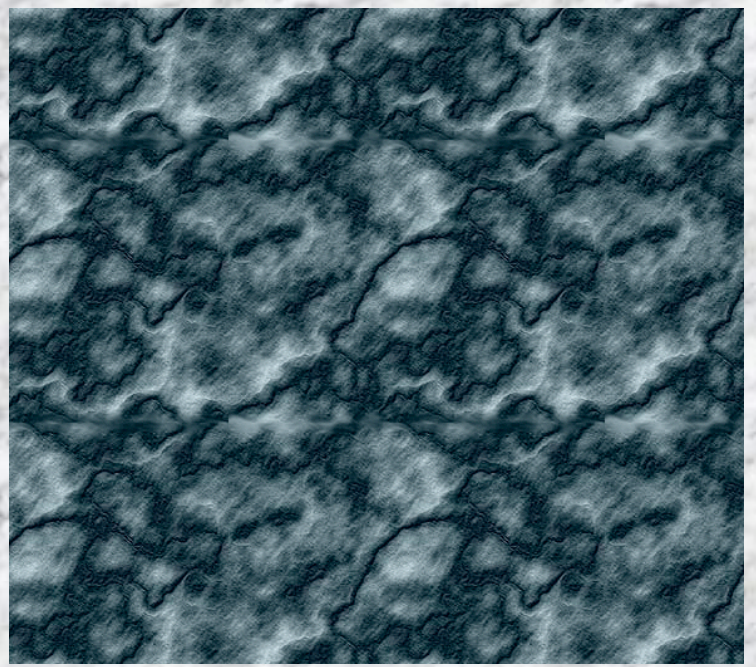
Texture scale

- Macro-texture :

A collection of objects (more or less variable) with a more or less regular repartition.

- Micro-texture :

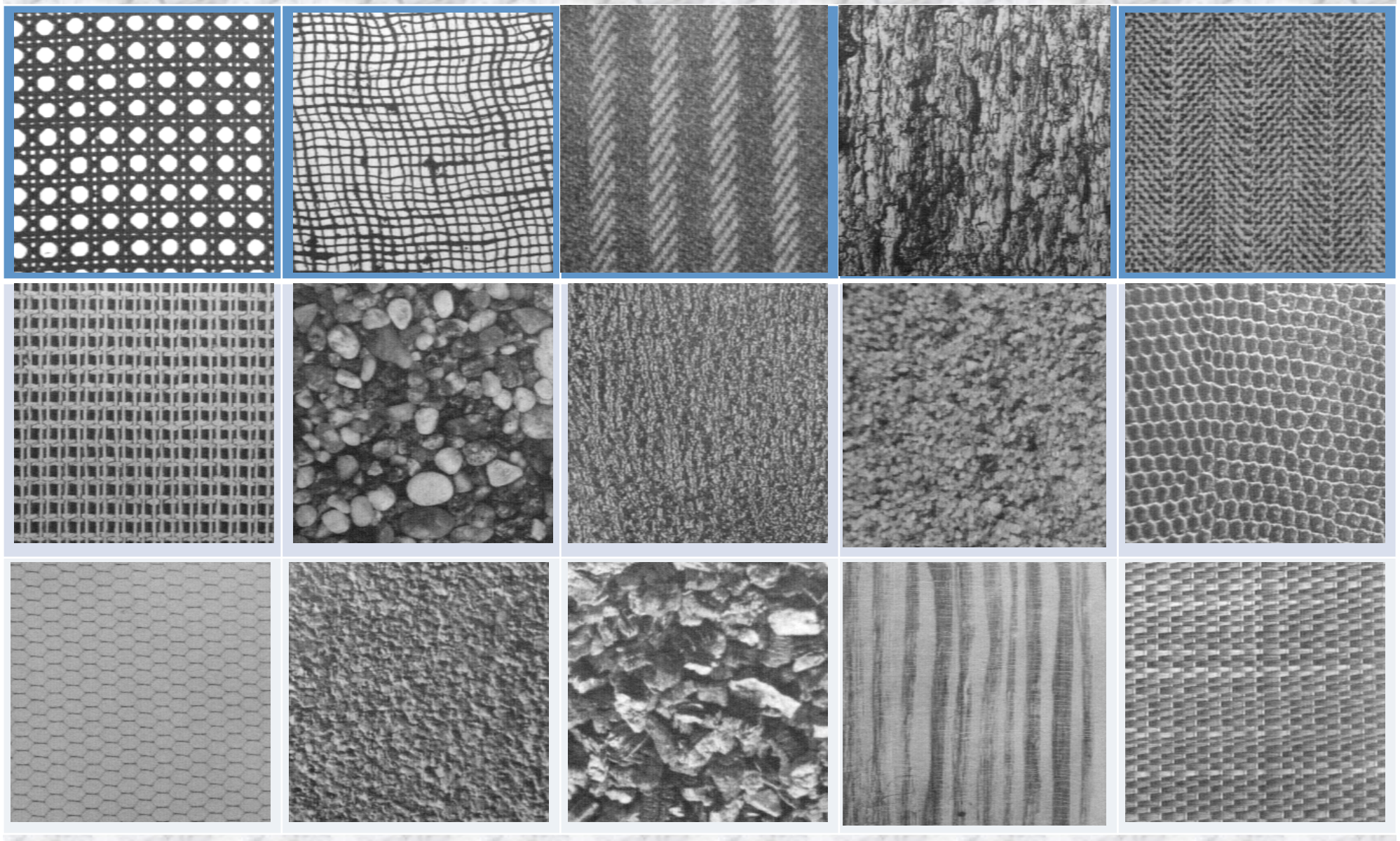
Realisation of some random phenomenon (random field)



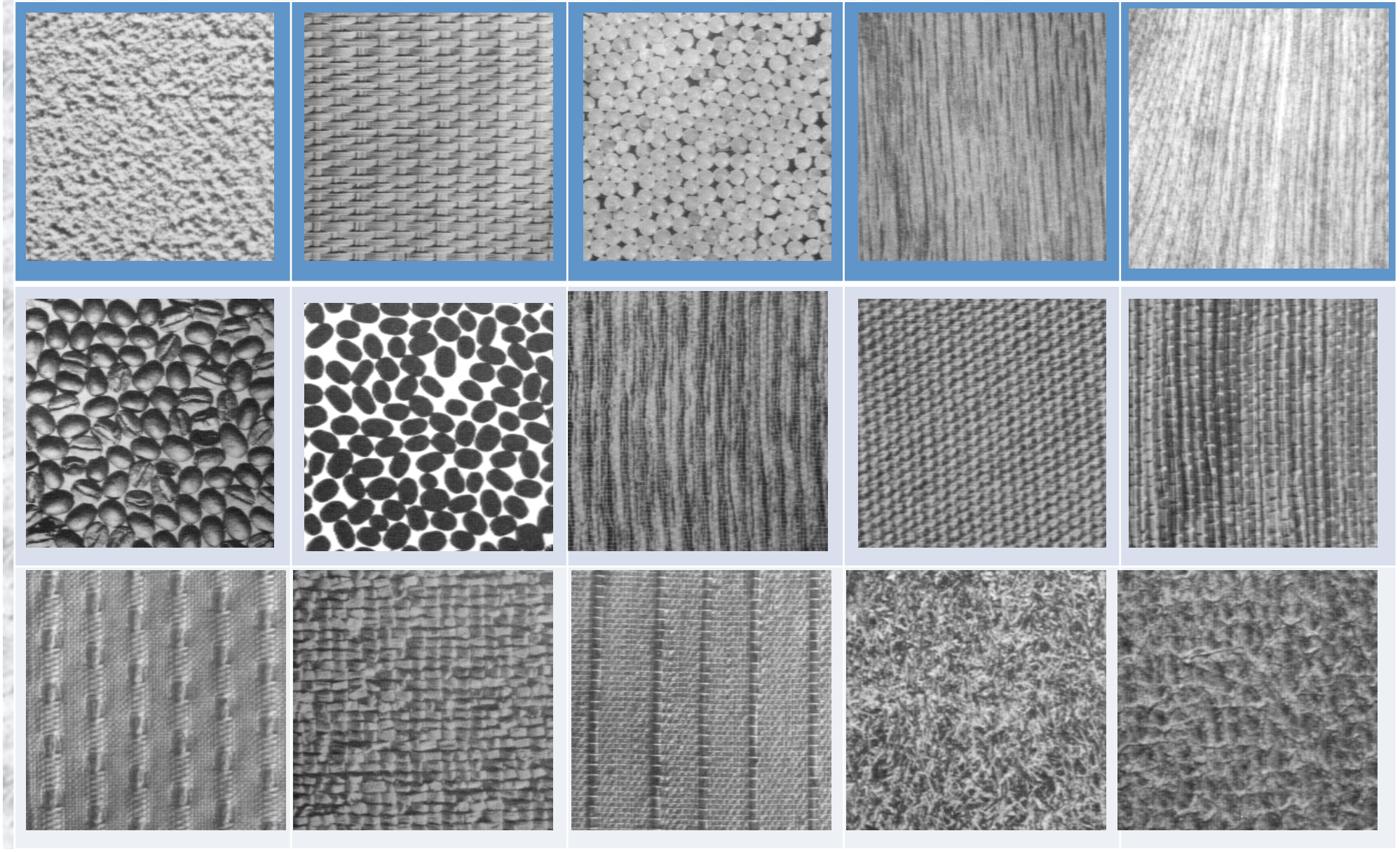
How to characterize a texture ?

- Synthesis vs Analysis
- Find periodicities :
  - Fourier Transform
- Find stationnary statistics :
  - Second order : cooccurrence matrices
- Try to mimic the visual system :
  - Filters (Gabor, …)
- Find global features :
  - Fractal dimension
- Find a representative basis :
  - Dictionnary of local patches
- Find a probabilistic model :
  - Markov Random Fields

Our data base : Brodatz album



Our data base : Brodatz album



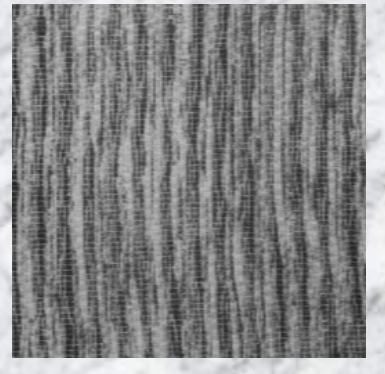
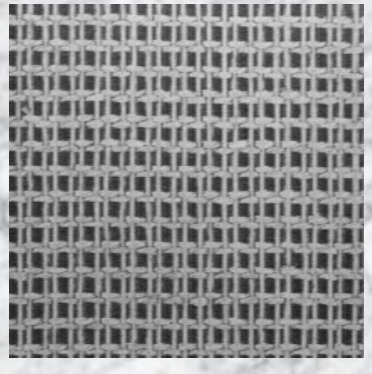
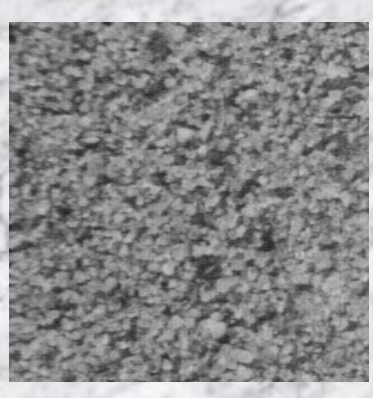
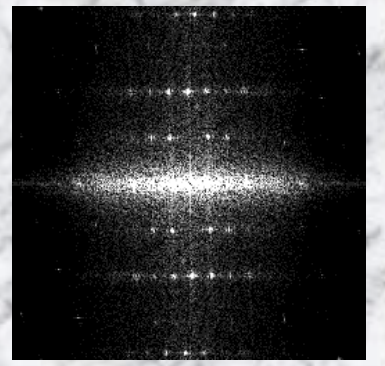
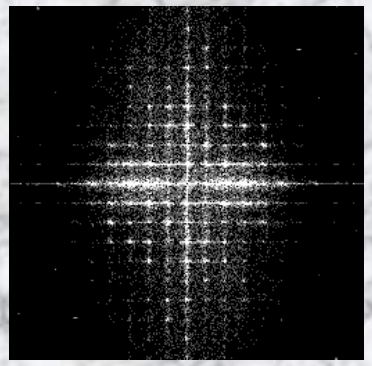
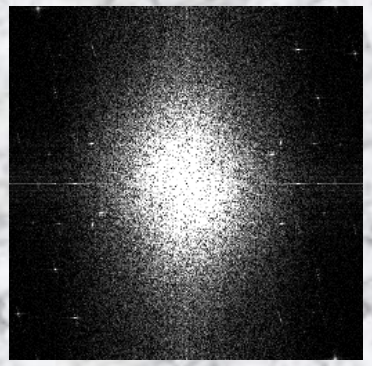
Fourier Transform

- To detect and characterize periodicities

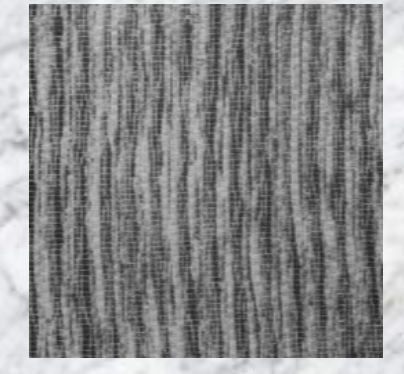
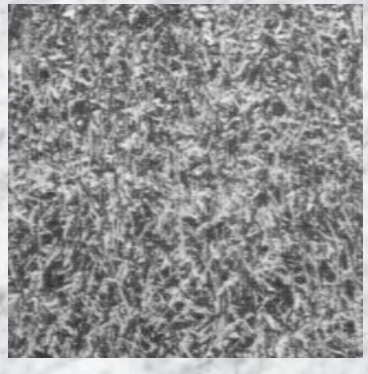
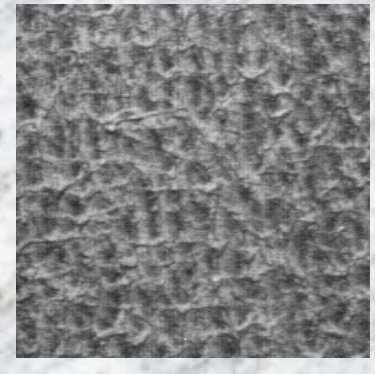
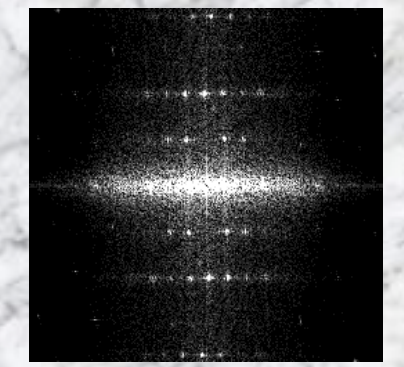
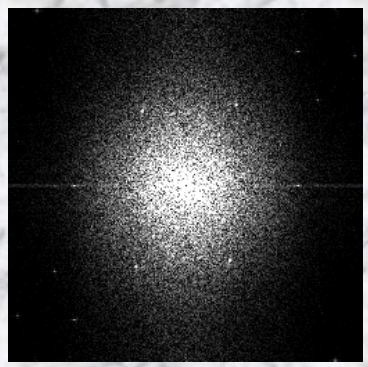
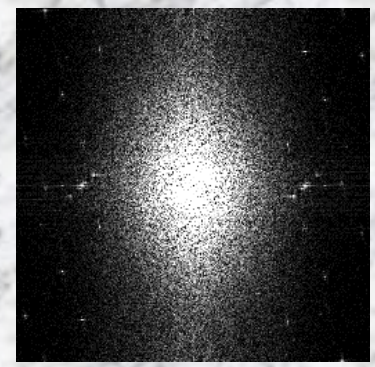
$$TF(u,v) = \iint f(x,y)\exp\left[-2\pi i(ux+vy)\right]dxdy$$

- Spectrum : $\left|TF(u,v)\right|^2$

Periodicities

Periodicities

Co-occurrence matrices

- Estimation of the joint probabilities

$$P\left(x_s = u, x_{s+\vec{d}} = v\right)$$

- Parameter : offset

$$\vec{d} = \left(\Delta i, \Delta j\right)$$

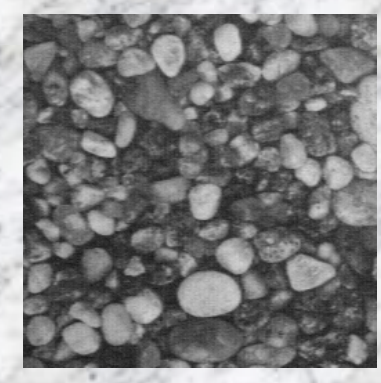
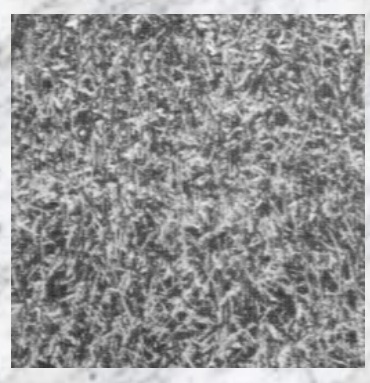
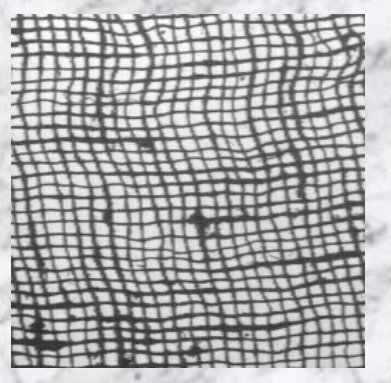
- Computation :

$$COOC_{\vec{d}}(u,v) \propto Card\left\{(i,j) : f(i,j) = u, f\left(i+\Delta i, j+\Delta j\right) = v\right\}$$

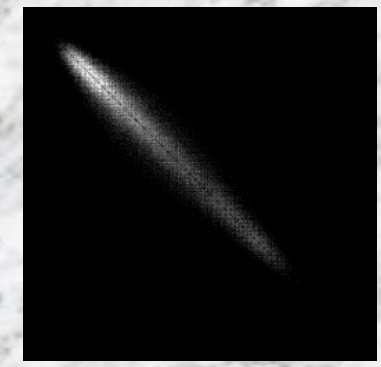
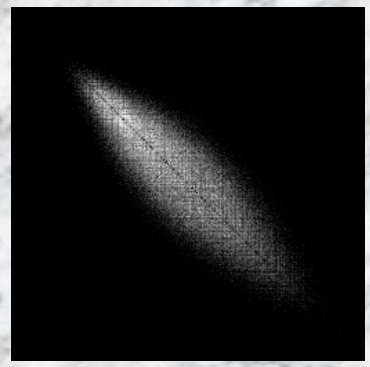
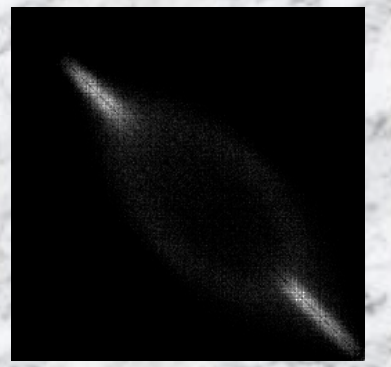
- For symmetry :

$$\frac{1}{2}\left(COOC_{\vec{d}} + COOC_{-\vec{d}}\right)$$

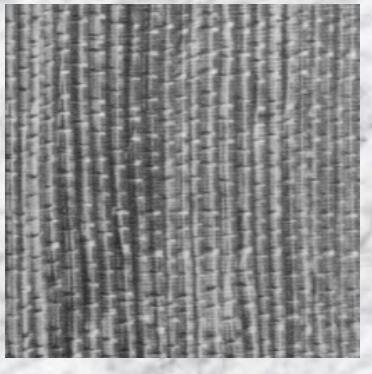
Co-occurrence and Correlation

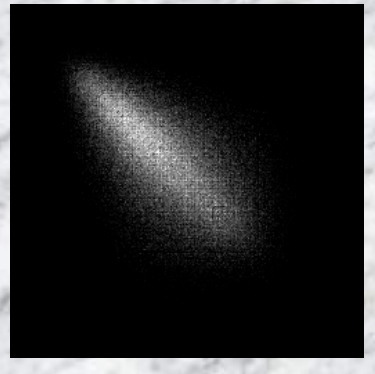




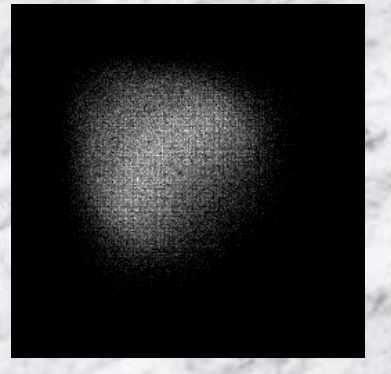
$$\Delta i = 0, \Delta j = 1$$

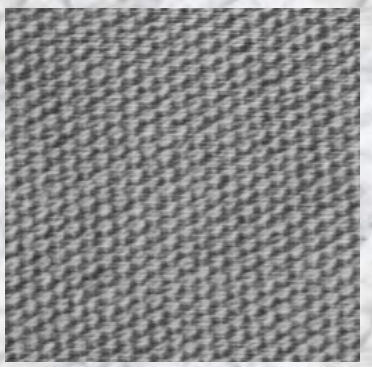
Co-occurrence and Correlation

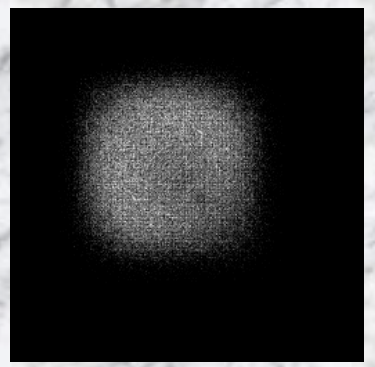


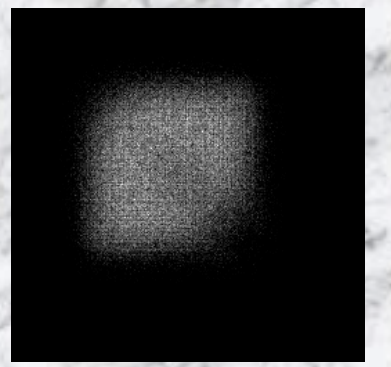




$\Delta i = 5, \Delta j = 0$ $\Delta i = 0, \Delta j = 5$



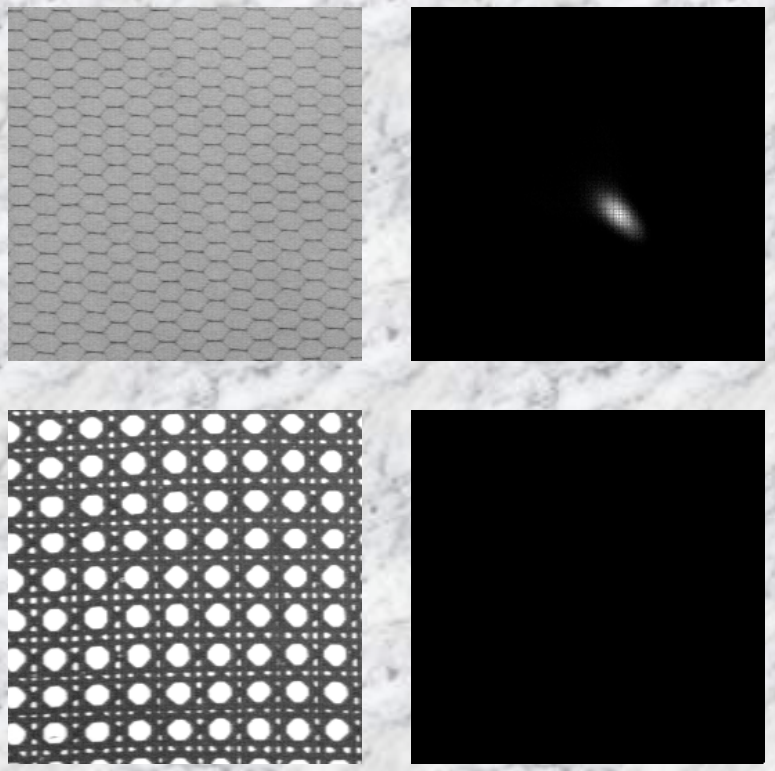




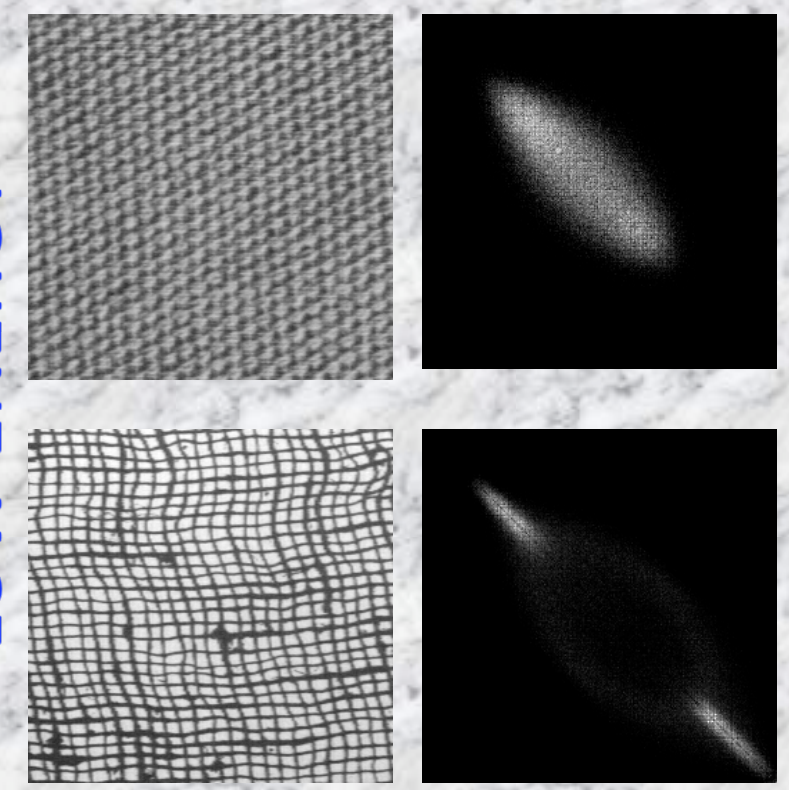
Co-occurrence : Energy

$$Energy = \sum_{(u,v)} \left(COOC(u,v)\right)^2$$

HIGH ENERGY



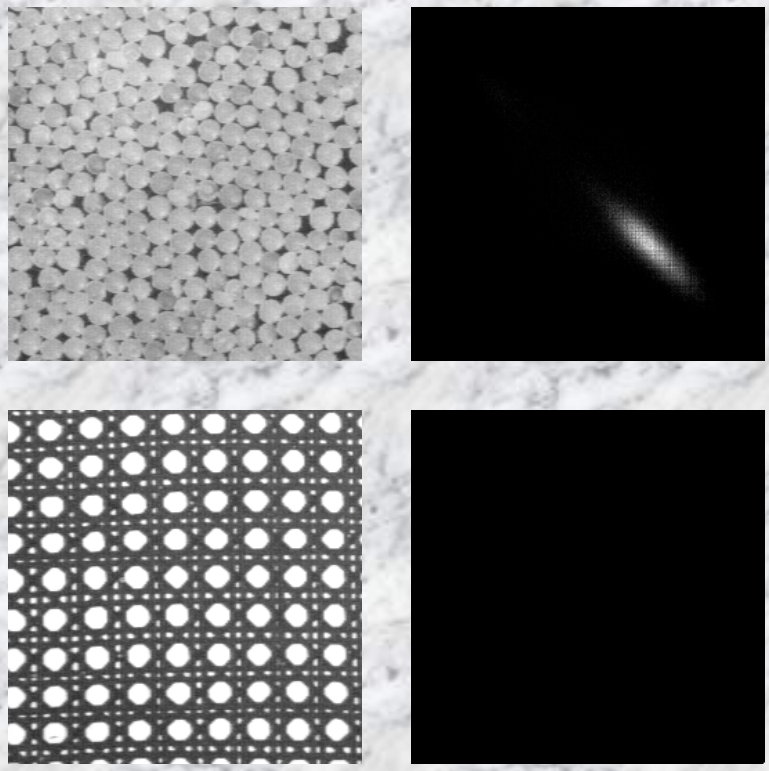
LOW ENERGY



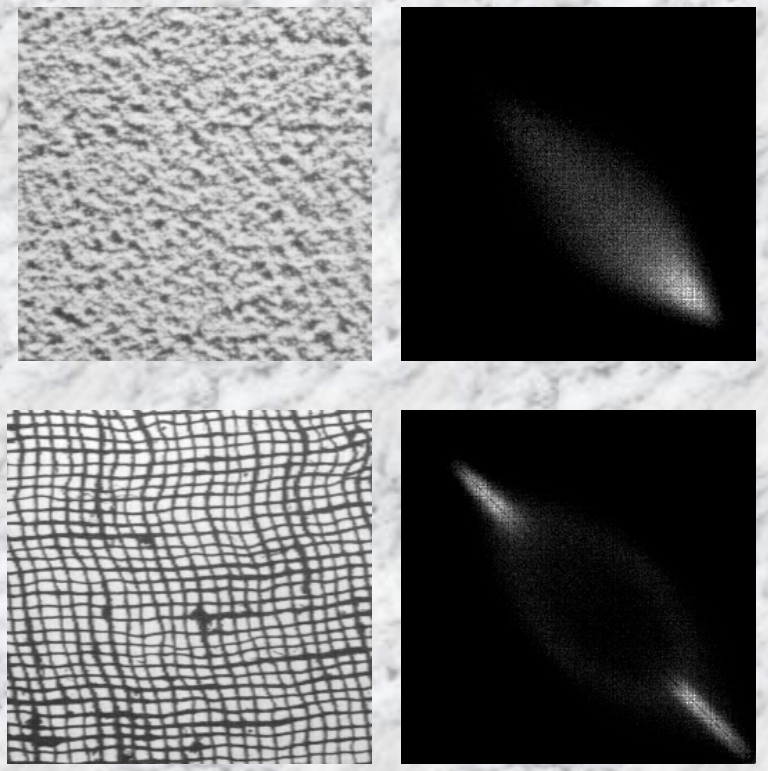
Co-occurrence : Entropy

$$Entropy = -\sum_{(u,v)} COOC(u,v)\log COOC(u,v)$$

LOW ENTROPY



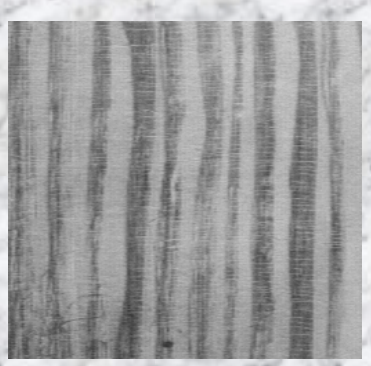
HIGH ENTROPY

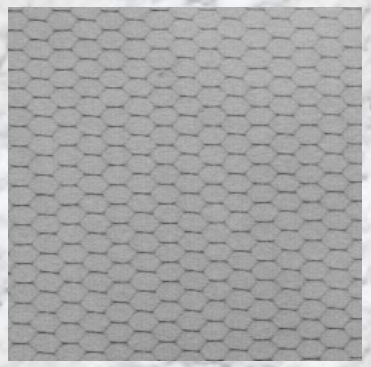


Co-occurrence : Correlation

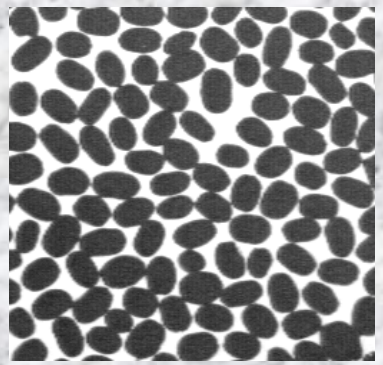
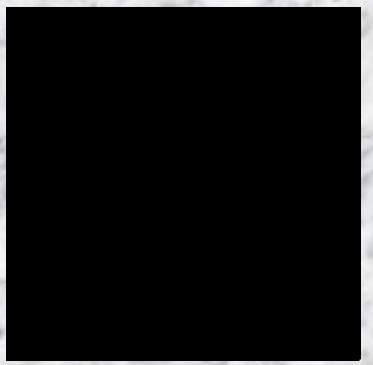
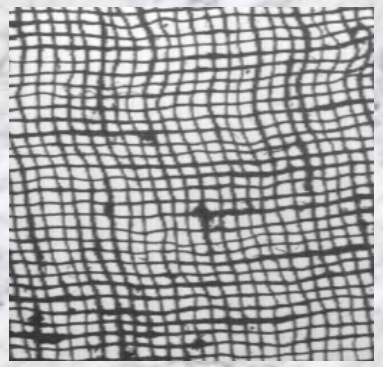
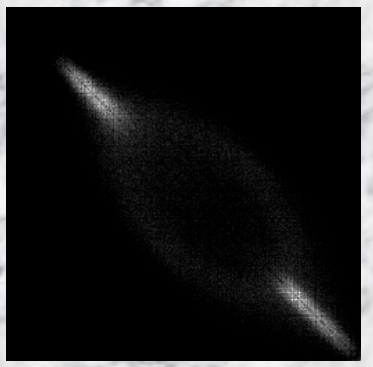
$$Correlation = \sum_{(u,v)} \frac{(u-\mu_u)(v-\mu_v)COOC(u,v)}{\sigma_u \sigma_v}$$

HIGH CORRELATION





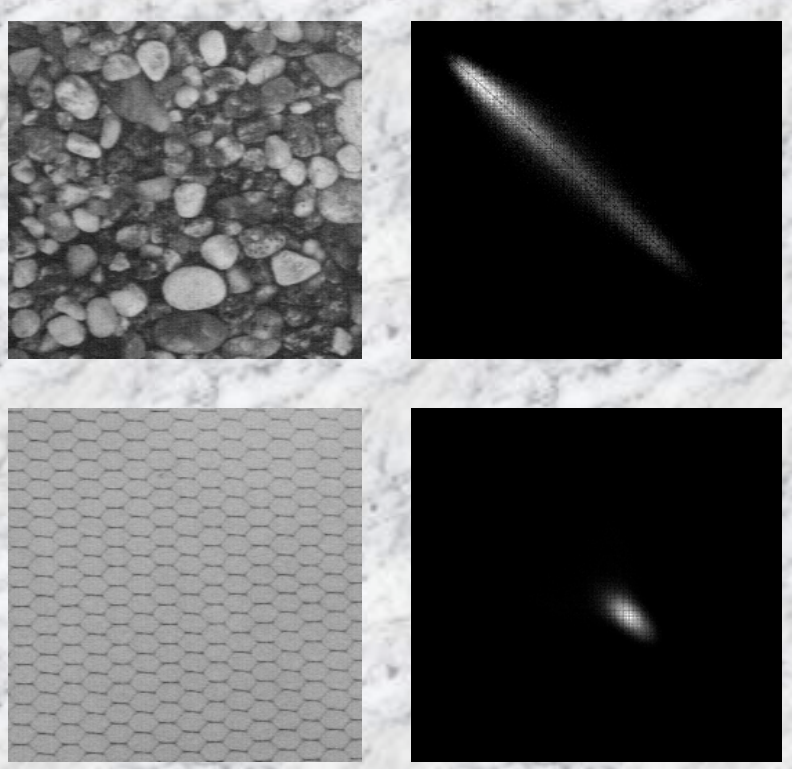

LOW CORRELATION

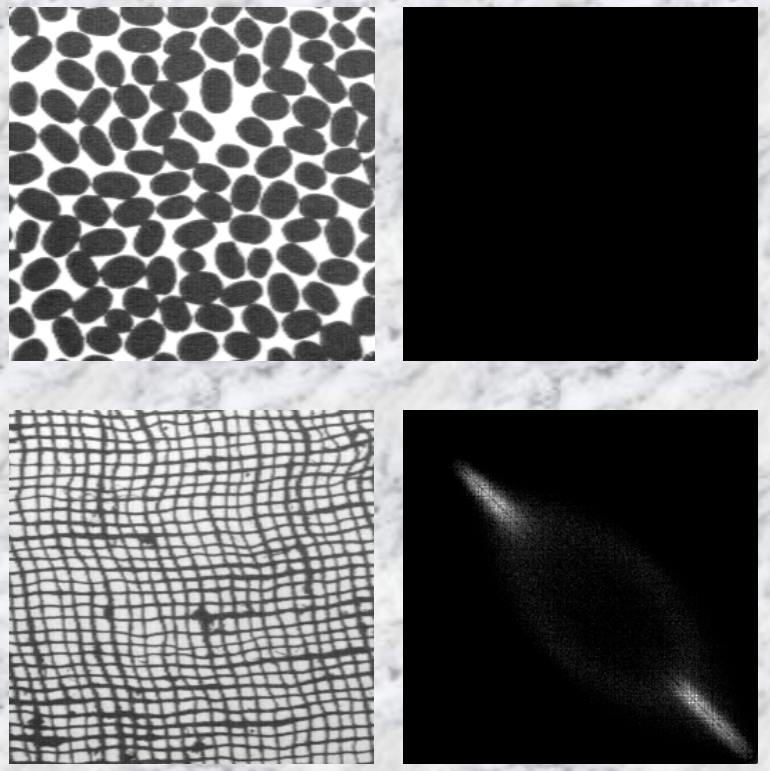
Co-occurrence : Inertia

$$Inertia = \sum_{(u,v)} (u-v)^2 \left(COOC(u,v)\right)^2$$

LOW INERTIA



HIGH INERTIA



And many more

- Inverse different moment $\sum_{(u,v)} \frac{1}{1+(u-v)^2} \left(COOC(u,v)\right)^2$
- Cluster shade $\sum_{(u,v)} \left((u-\mu_u)+(v-\mu_v)\right)^3 \left(COOC(u,v)\right)^2$
- Cluster prominence $\sum_{(u,v)} \left((u-\mu_u)+(v-\mu_v)\right)^4 \left(COOC(u,v)\right)^2$
- ……

Gabor Filters

- Study the similarity between the signal locally and a bank of filters
- Gabor filter : sinusoidal shape weighted by a Gaussian
- Parameters :
  - Frequency
  - Orientation
  - Phase
  - Gaussian variance

$$\begin{cases} \lambda \\ \theta \\ \phi \\ \sigma^2 \end{cases}$$

Gabor filters

$$G(x,y \mid \lambda,\theta,\phi,\sigma^2,X,Y) = \exp\left[-\frac{(x-X)^2+(y-Y)^2}{2\sigma^2}\right]\sin\left(\lambda\left(x\cos\theta - y\sin\theta\right)+\phi\right)$$

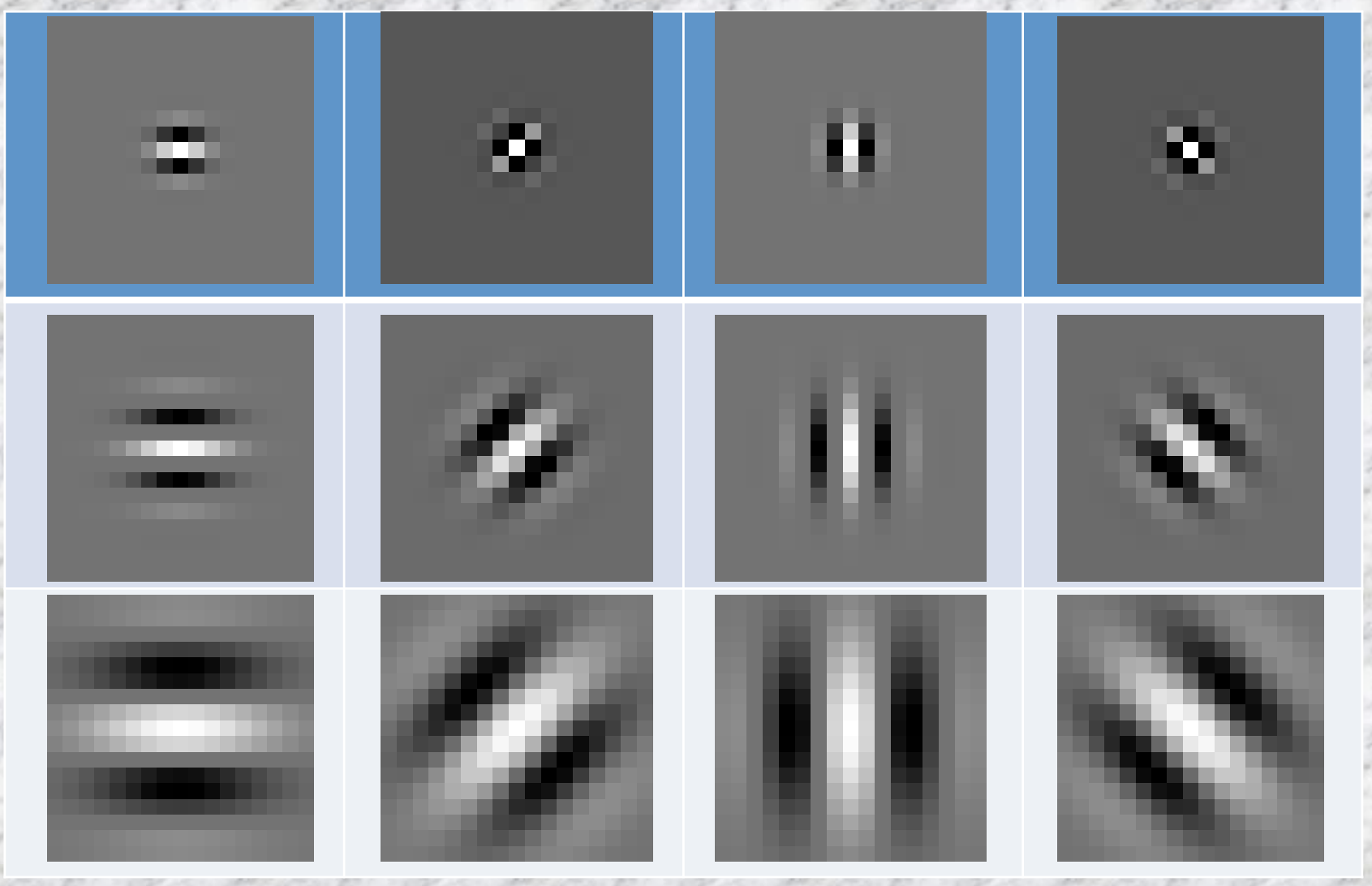
Energy :

$$S(X,Y) = \left(\sum_{x,y} G(x,y \mid \lambda,\theta,0,\sigma^2 X,Y) I(x,y)\right)^2 + \left(\sum_{x,y} G\left(x,y \mid \lambda,\theta,\frac{\pi}{2},\sigma^2 X,Y\right) I(x,y)\right)^2$$

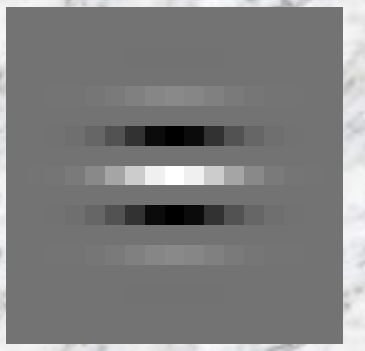
In practice :

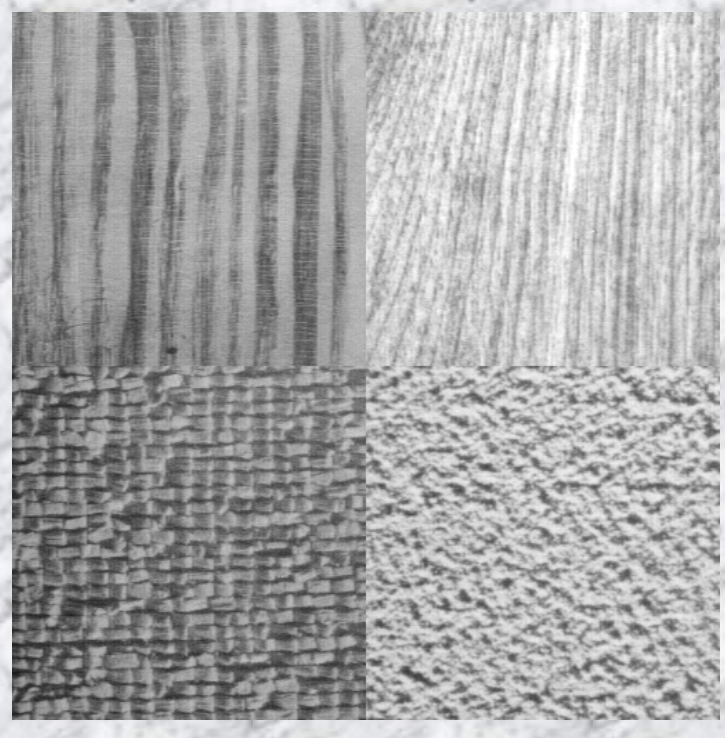
$$F(X,Y) = \max_{\phi}\left(\sum_{x,y} G(x,y \mid \lambda,\theta,\phi,\sigma^2 X,Y) I(x,y)\right)^2$$

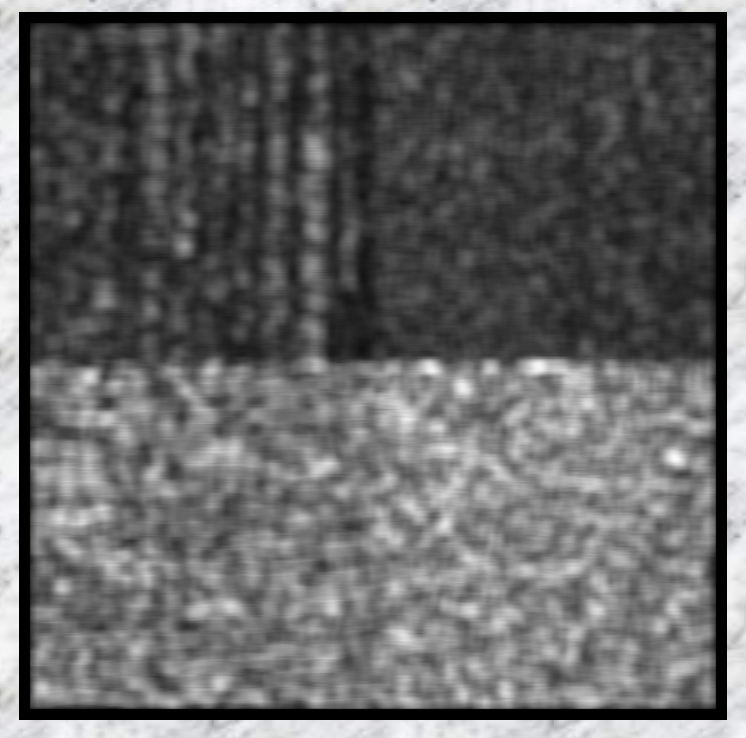
Gabor filters



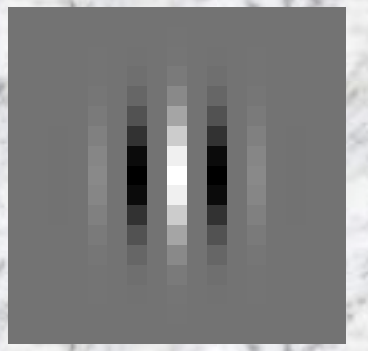
Gabors Filters

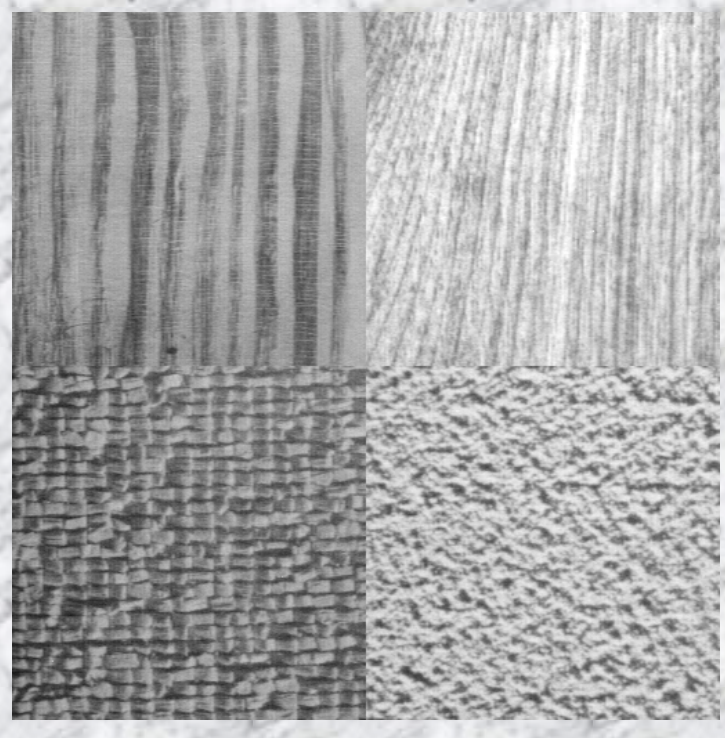


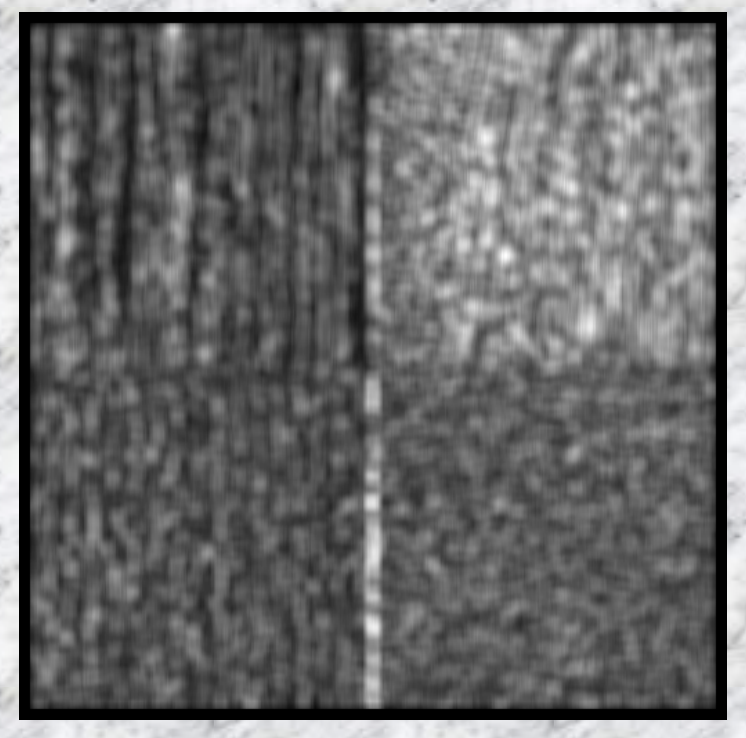




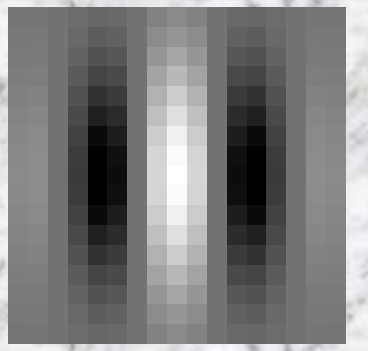
Gabors Filters

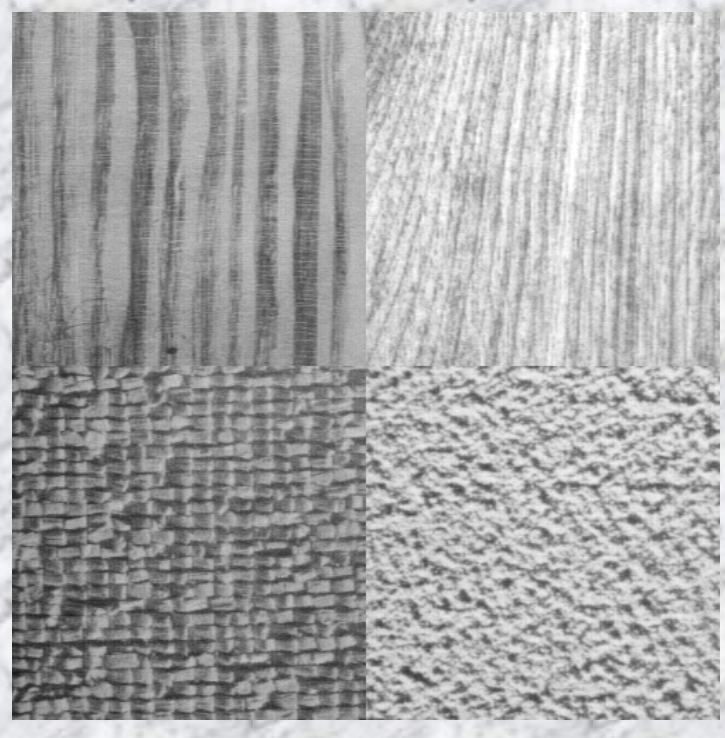


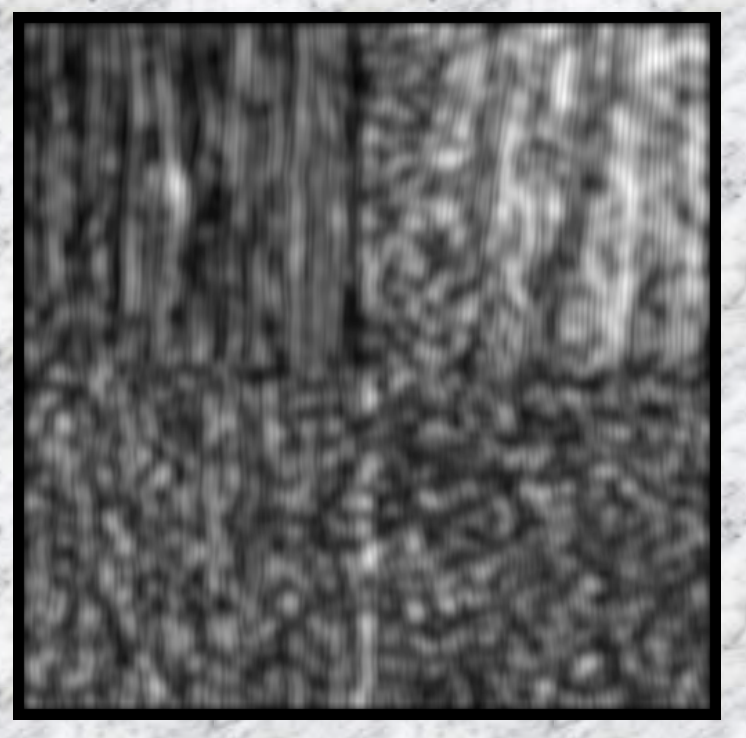




Gabors Filters



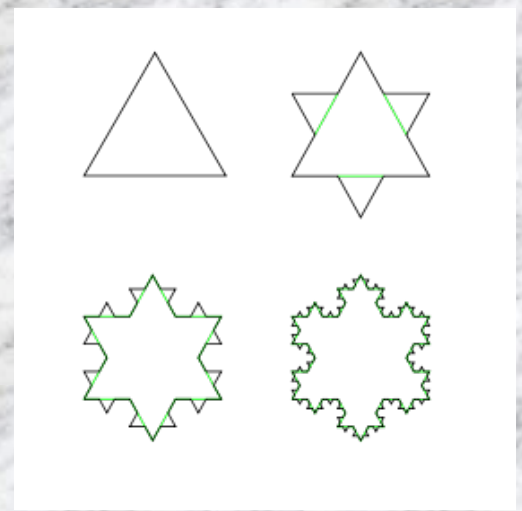




Fractal dimension

Generalization of Mendelbrot idea to describe the Britany coast :

- scale invariance : at each scale the « smoothness » is the same
- well-known example : the Korch snowflake



$$l_n = \left(\frac{4}{3}\right)^n l_0 \xrightarrow[n\to\infty]{} \infty, \quad S_n < 3S_0 \text{ finite}, \quad \mathrm{D_F} = \frac{\log 4}{\log 3} \approx 1.26$$

- Objects having fractal dimensions : lungs, capilaries,….

- Idea : compute the fractal dimension of the texture surface

The counting box method

Consider an object A composed of N_r replica of the same object with a scale factor r, then A is **self-similar** and its fractal dimension is given by :

$$D = \frac{\log N_r}{\log r^{-1}}$$

In practice, we estimate N_r for several values of r and compute the slope of the curve log N_r as a function of log r^{-1} by a linear regression.

N_r estimation :

For a cube of size r x r x αr

We consider the cells C(i,j) = [ir,(i+1)r] x [jr,(j+1)r]

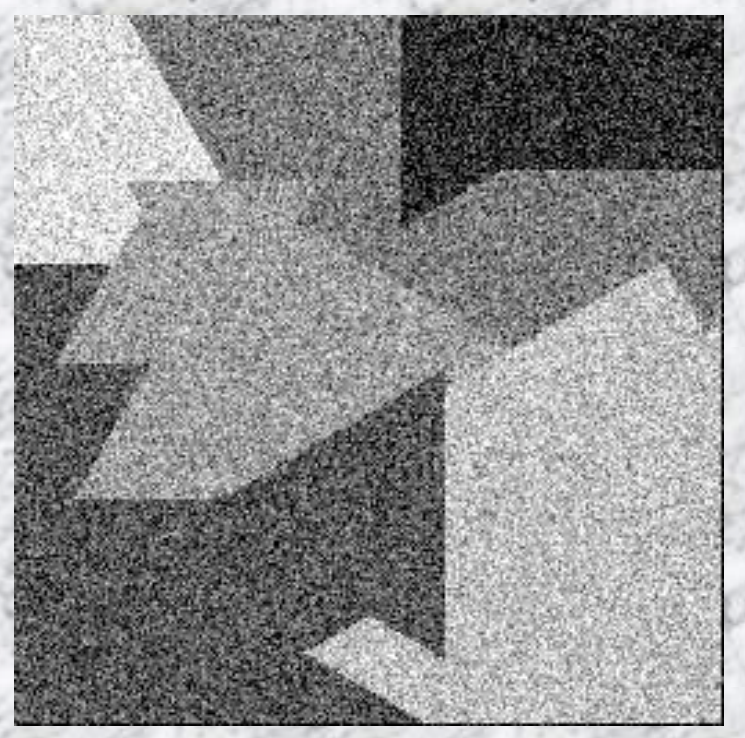
Then :

$$N_r(i,j) = ceil\left(\frac{\max_{C(i,j)}\{I(i,j)\} - \min_{C(i,j)}\{I(i,j)\}}{\alpha r}\right)$$

$$N_r = \sum_{i,j} N_r(i,j)$$

The counting box method

$D_F = 2.05$



$D_F = 2.80$

Texture dictionnary

For each pixel, we can define a patch of size n x n (local window) :

$$P_{x,y}(I)(d_x, d_y) = I(x + d_x, y + d_y)$$

-> To each image corresponds a set of patches : $\left\{P_{x,y}(I),\ 1 \le x \le NL,\ 1 \le y \le NC\right\}$

Considering a set of patches $\left\{p_{x,y},\ 1 \le x \le NL,\ 1 \le y \le NC\right\}$, we can reconstruct an image :

$$\hat{I} = \arg\min_{I} \sum_{x,y} \left\| P_{x,y}(I) - p_{x,y} \right\|^2$$

By averaging on the patches that overlap a given pixel :

$$\hat{I}(x,y) = \frac{1}{n^2} \sum_{(i,j):\begin{cases} |i-x| \le n/2 \\ |j-y| \le n/2 \end{cases}} p_{i,j}(x - i, y - j)$$

Texture dictionnary

Consider a set of m normalized patches (non localized) : $D = \left(d_1,\ldots,d_{k,}\ldots d_m\right)$

D is a dictionnary (matrix) containing m atoms

Consider an image (a texture) I, for each pixel (x,y) we can estimate the « best » linear combination of atoms to represent I :

$$p_{x,y} = \sum_{k=1}^{m} w_k^{x,y} d_k \text{ such that } \left\{w_k^{x,y}\right\} = \arg\min \left\| p_{x,y} = \sum_{k=1}^{m} w_k^{x,y} d_k \right\|^2$$

We can then reconstruct I : $\hat{I}(x,y) = \frac{1}{n^2} \sum_{(i,j):\begin{cases}|i-x|\le n/2\\|j-y|\le n/2\end{cases}} p_{i,j}(x-i,y-j)$

And compute a reconstruction error : which depends on the dictionnary

$$\sum_{x,y} \left\| P_{x,y}\left(\hat{I}\right) - P_{x,y}(I) \right\|^2 \text{ or } \left\| \hat{\mathrm{I}} - \mathrm{I} \right\|^2$$

Texture dictionnary

Consider we have N dictionnaries associated with N known textures,
If we have an unknown texture we can compute its distance (reconstruction error)
With respect to the N dictionnaries, and thus perform a classification.

Question : How to construct a dictionnary with a reasonable number of atoms (sparsity constraint) ?

This is an optimization problem :

$$\min \left\| P\left(I\right) - w\ D \right\|^2 \text{ such that } \left\| \mathrm{w} \right\|_0 \leq \mathrm{L}$$

We have to estimate w and D. For simplicity we solve it alternatively with respect to w and D and iterate the process.

Optimization with respect to w : Orthogonal Matching Pursuit (OMP)

We can write : $P_{x,y}(I) = a_s = \langle a_s, d_j \rangle d_j + R_{l/j}$

Where : $d_j \perp R_{l/j}$ So we have :

Algorithm :

1) Minimize : $j_i = \arg\min_j \langle a_s, d_j \rangle$

2) Update :
and go to 1) $a_s^{(i+1)} = a_s^{(i)} - \langle a_s^{(i)}, d_{j_i} \rangle d_{j_i}$

We can fix the maximum number of atoms or iterate untill the residual is small enough

Optimization with respect to D : Method of Optimal Direction (MOD)

We search : $\tilde{d}_j = d_j + \delta_j$ such that $\sum_l \left\|\tilde{R}_l\right\|^2 < \sum_l \left\|R_l\right\|^2$

$$\begin{cases} R_l = a_l - \sum_{j=1}^{k} w_{l,j} d_j \\ \tilde{R}_l = R_l - \sum_{j=1}^{k} w_{l,j} \delta_j \end{cases}$$

Optimization :

$$\arg\min_{\delta} \left\{ \sum_l \left\|\tilde{R}_l\right\|^2 = \sum_l \left\|R_l\right\|^2 - A \right\} = \arg\max_{\delta} \left\{ 2 \sum_l \sum_{j=1}^{k} w_{l,j} \delta_j^t R_l - \sum_l \sum_{j=1}^{k} \sum_{i=1}^{k} w_{l,j} w_{i,j} \delta_j^t \delta_i \right\}$$