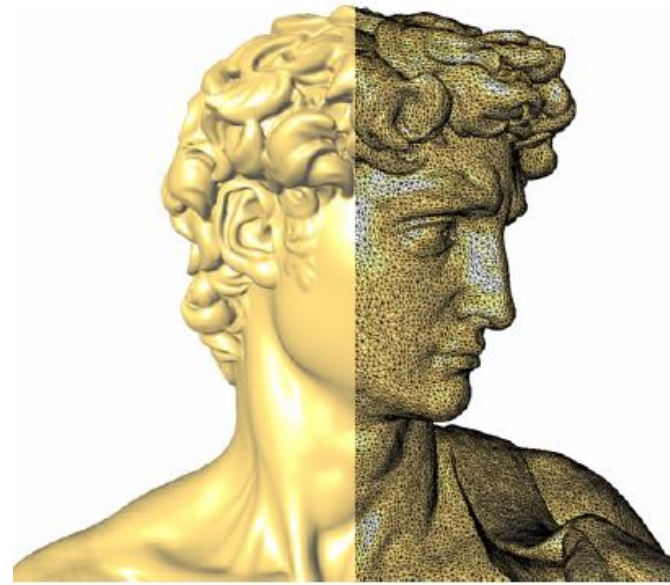
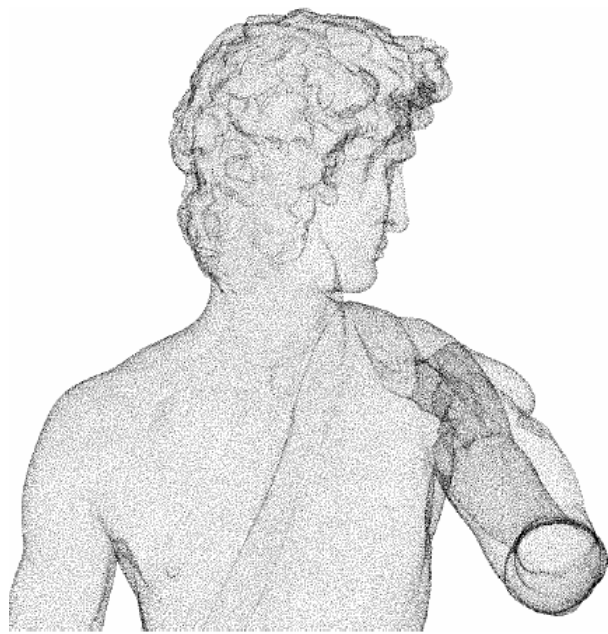


# Algorithmique géométrique appliquée à l'image



# Diagramme de Voronoi domaines d'applications

Image et informatique graphique

Imagerie médicale

Biologie

Réseaux

Base de données

Géographie

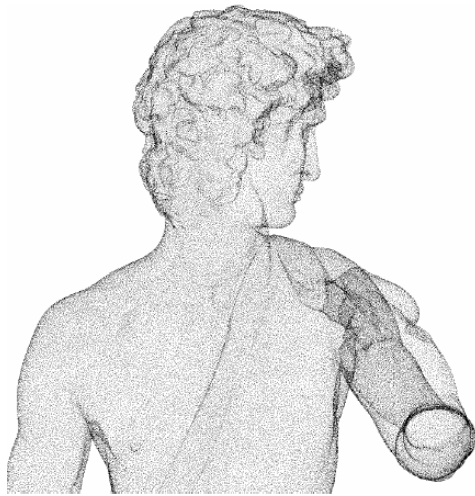
Astronomie

Sport

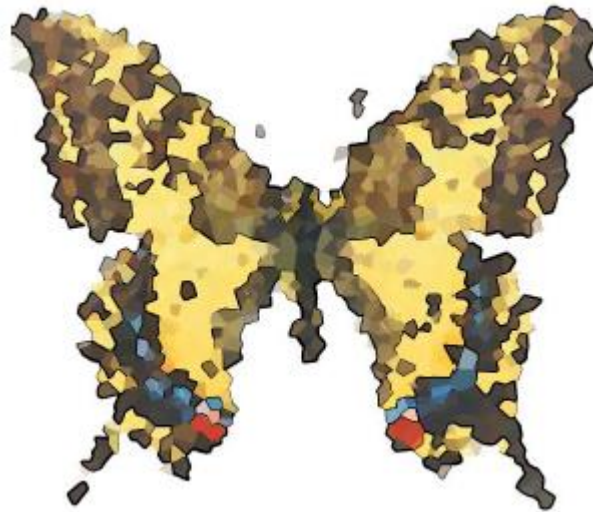
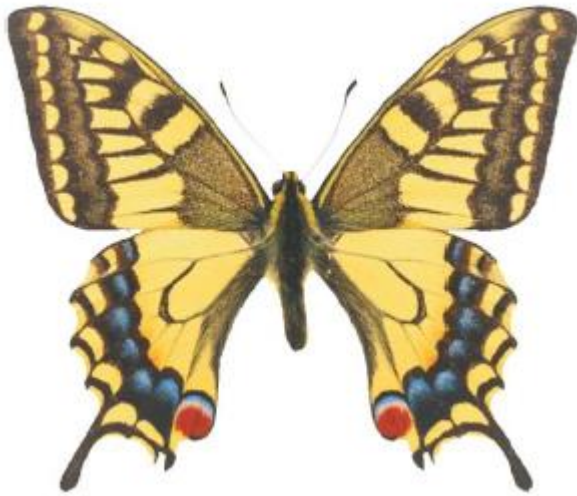
Art

.....

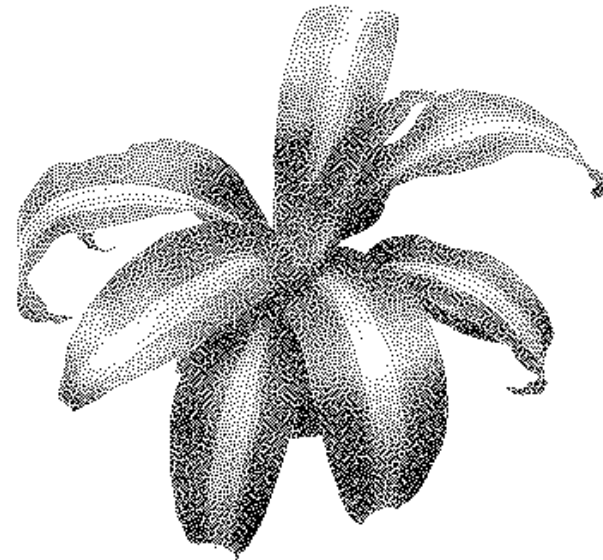
# Exemples



Reconstruction



Effet mosaïque



Effet pointillé  
« Stippling »



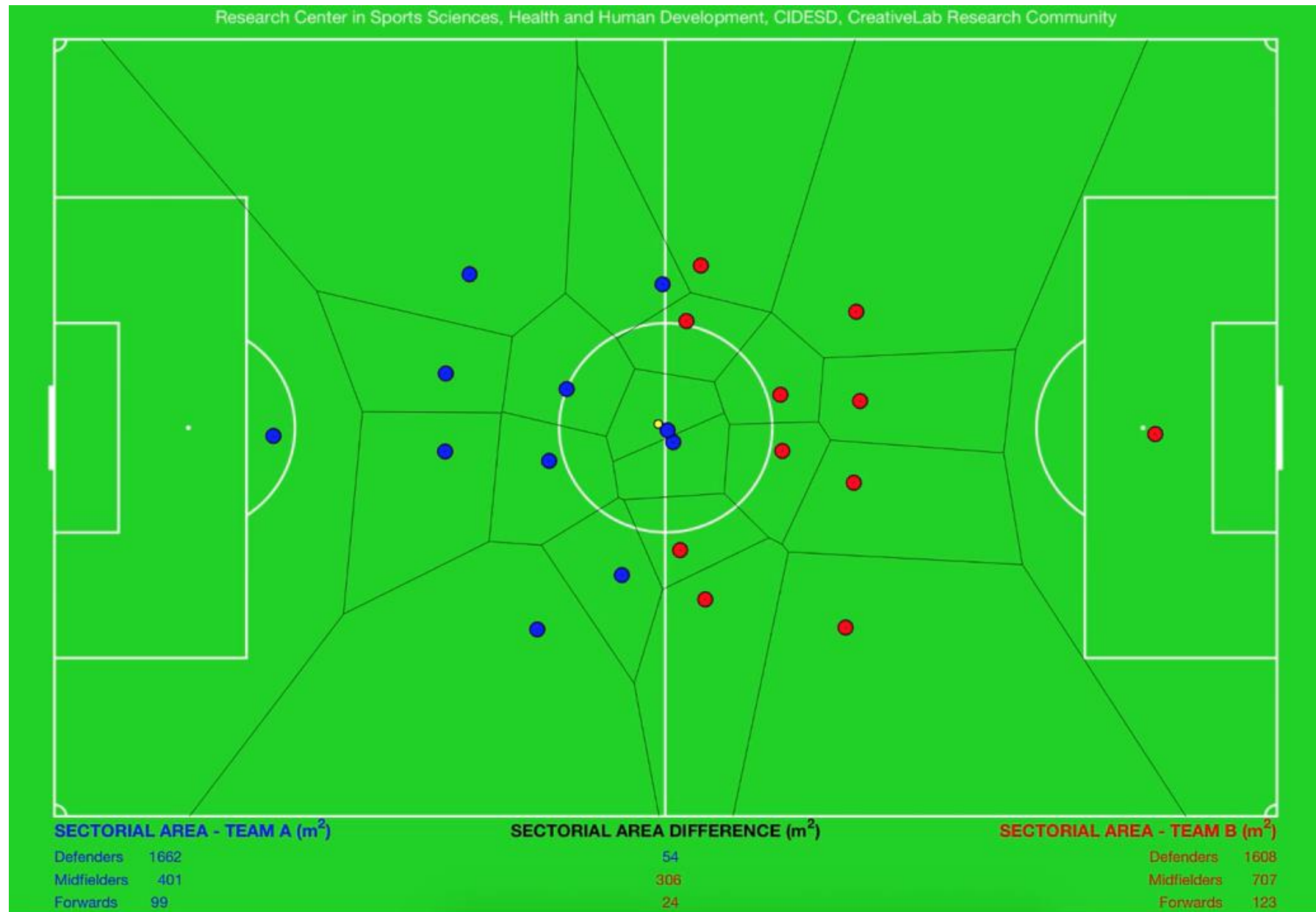
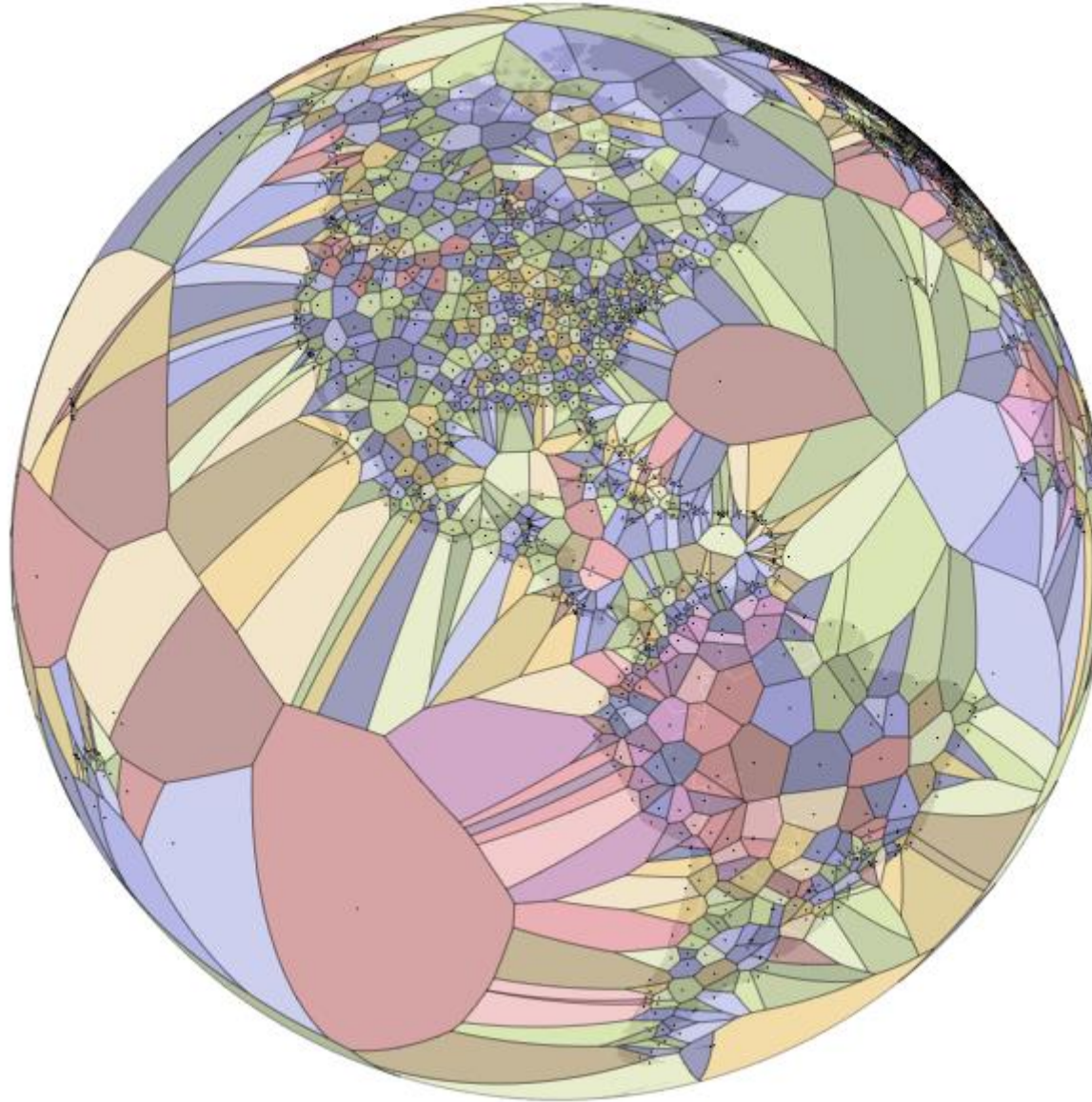
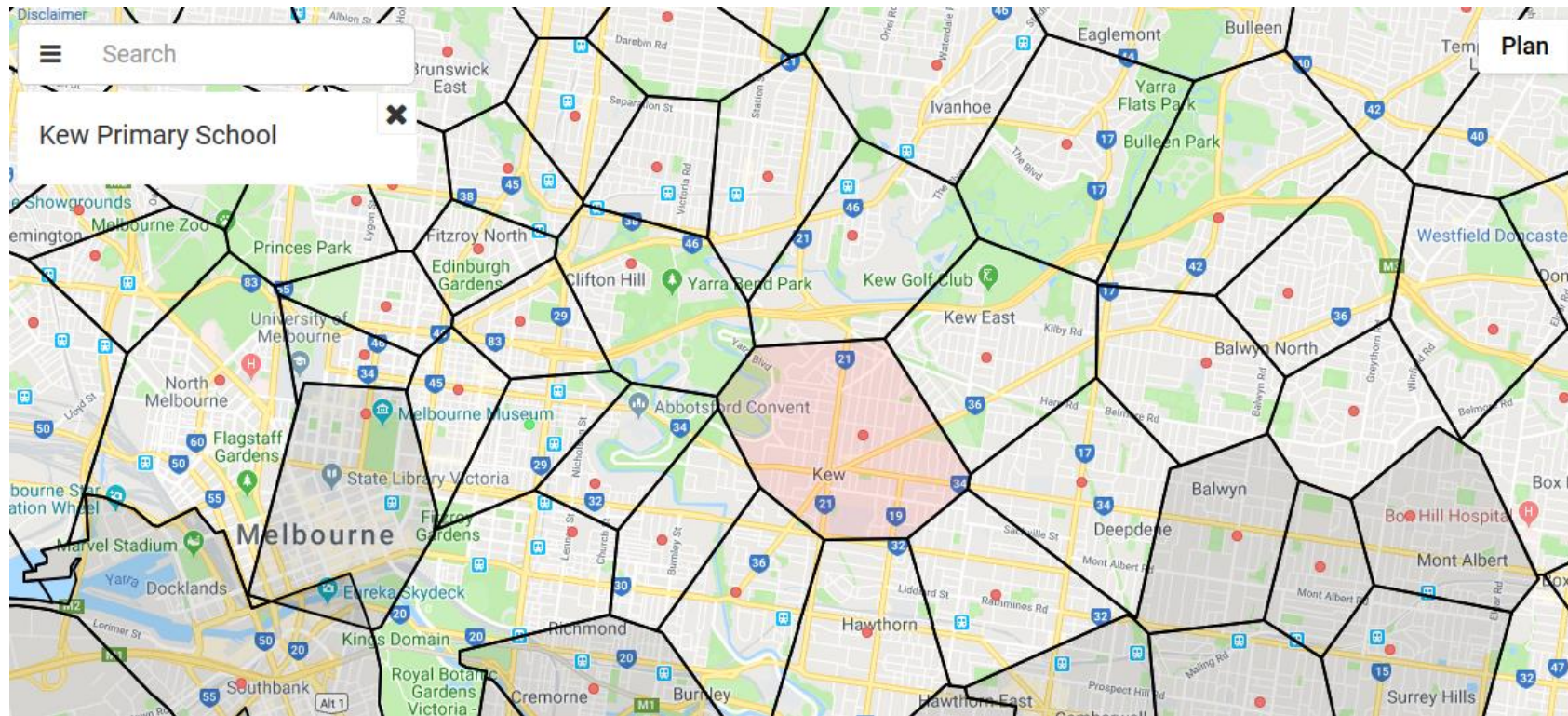


Diagramme de Voronoi appliquée au football



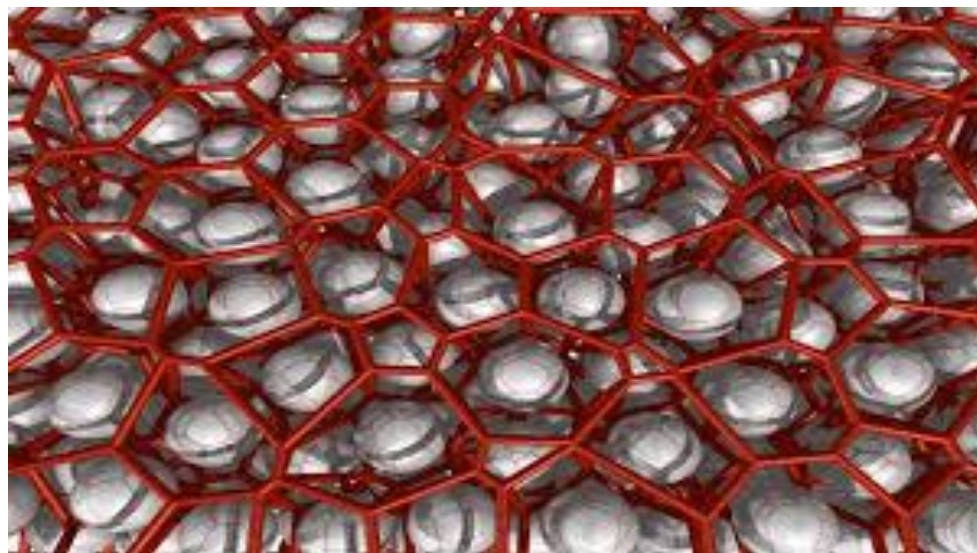
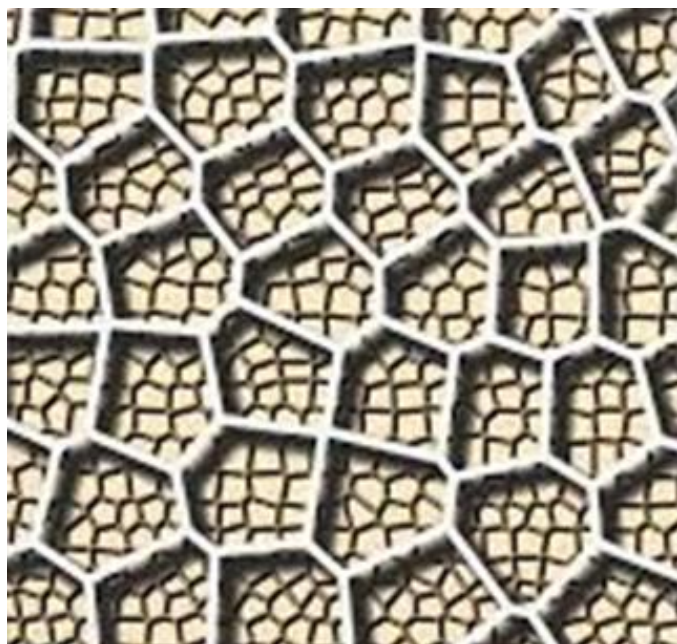




En Australie, les écoles publiques admettent les élèves éligibles à l'école primaire la plus proche de leur lieu de résidence. Les élèves et les parents peuvent voir les "écoles" les plus proches de leur résidence.

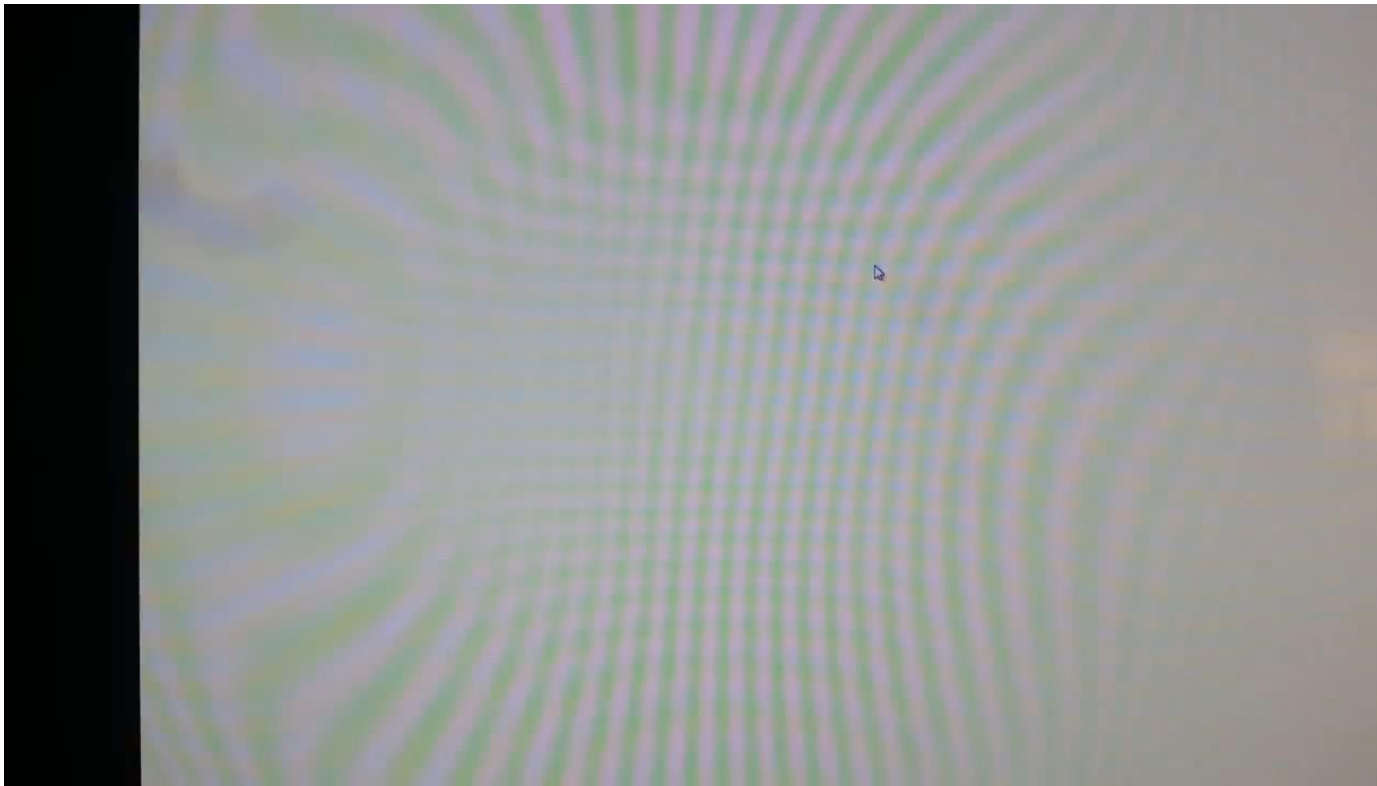
<http://melbourneschoolzones.com/schools/kew-primary-school/>



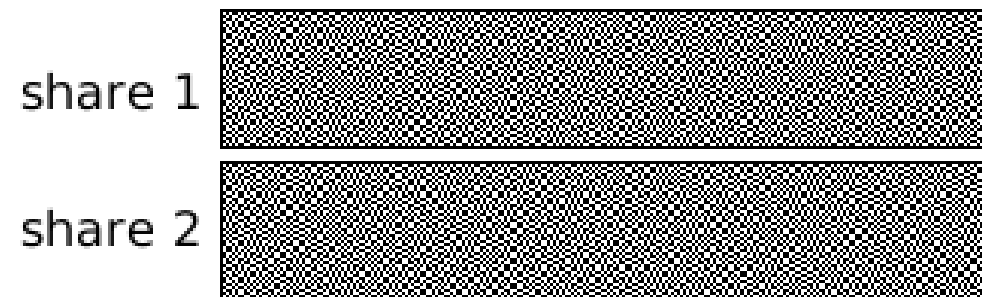


L'art Voronoi

## Exemples

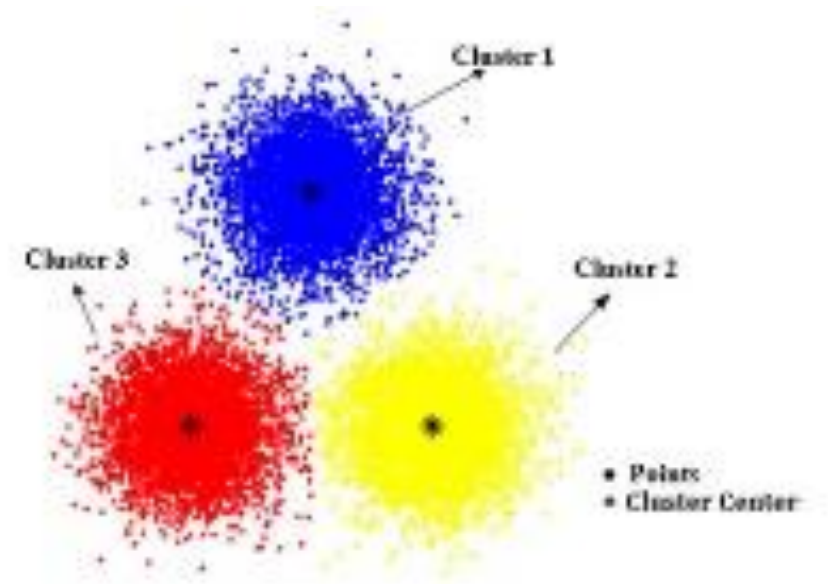


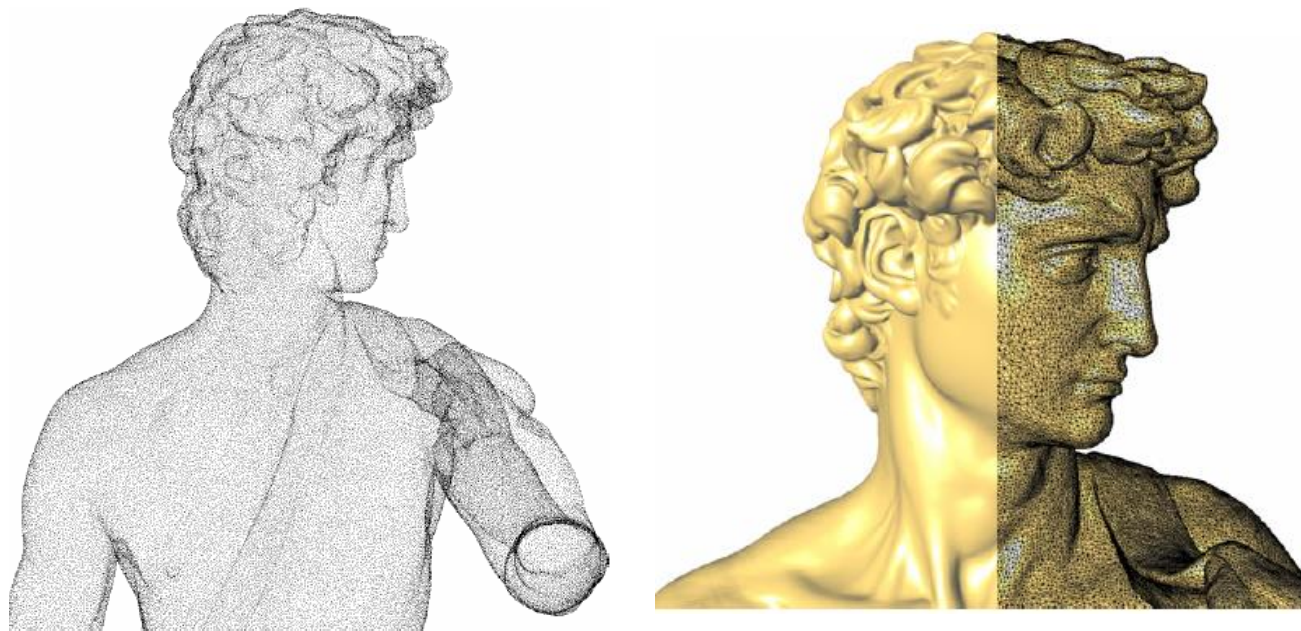
Alpha-complexe et alpha-shape



Cryptographie visuelle







## Diagramme de Voronoi, triangulation de Delaunay : algorithmes et applications

- ❖ Triangulation de Delaunay et diagramme de Voronoi
- ❖ Algorithme pour calculer Diagramme de Voronoi discret
- ❖ Approximation d'images, compression
- ❖ Morphing
- ❖ Segmentation : Algorithme K-means.
- ❖ Cryptographie visuelle



- Installer OpenCV :
  - Tutoriel vidéo (sur youtube)
  - Installation Cheat Sheet 1 - OpenCV 3 and C++.pdf
    - Guide : étapes décrites pas à pas en moins de 3 pages



## OpenCV 3 Windows Installation Guide

Chris Dahms · 4 vidéos · 12 366 vues · Dernière modification le 1 janv. 2016

▶ Tout regarder

◀ Partager

+ Enregistrer

1



OpenCV 3 Windows 10 Installation Tutorial - Part 1 - C++

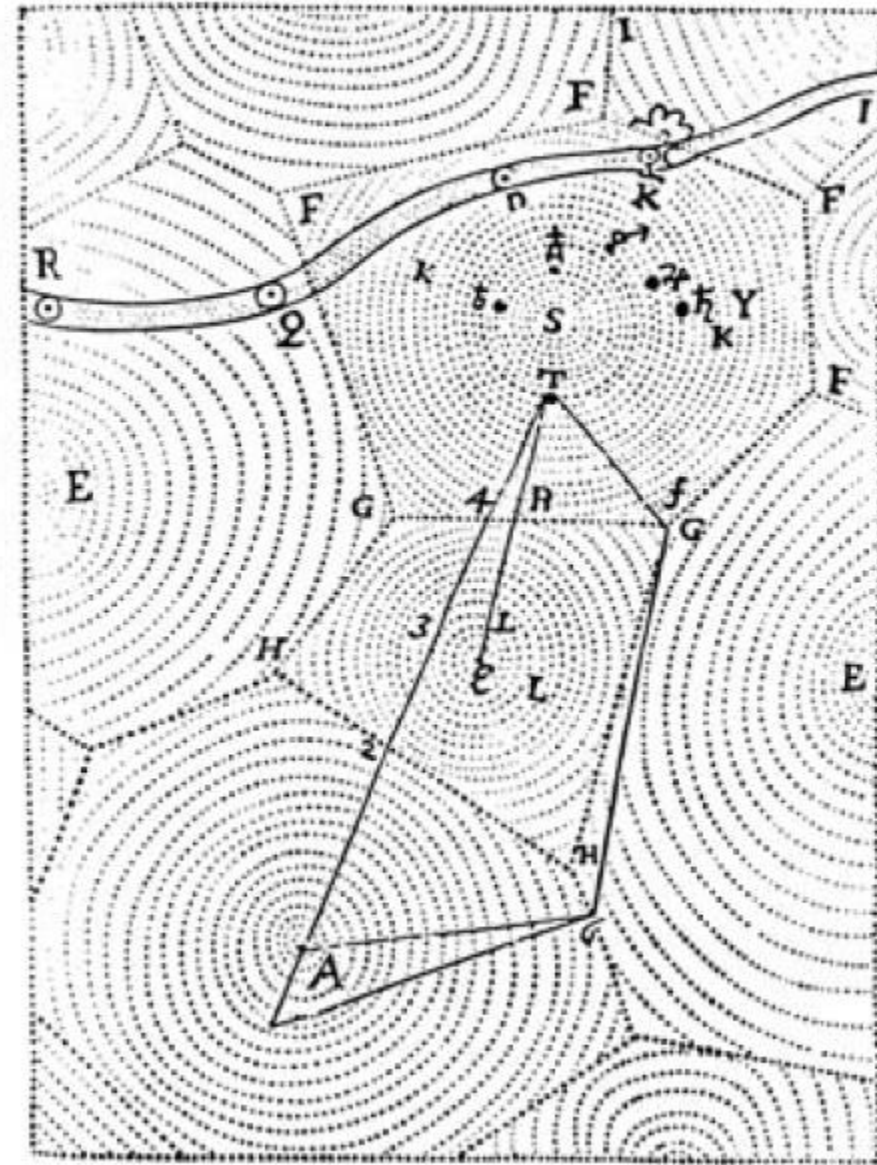
de Chris Dahms

Source : [https://github.com/MicrocontrollersAndMore/OpenCV\\_3\\_Windows\\_10\\_Installation\\_Tutorial](https://github.com/MicrocontrollersAndMore/OpenCV_3_Windows_10_Installation_Tutorial)

# 1. Diagramme de Voronoi et Triangulation de Delaunay

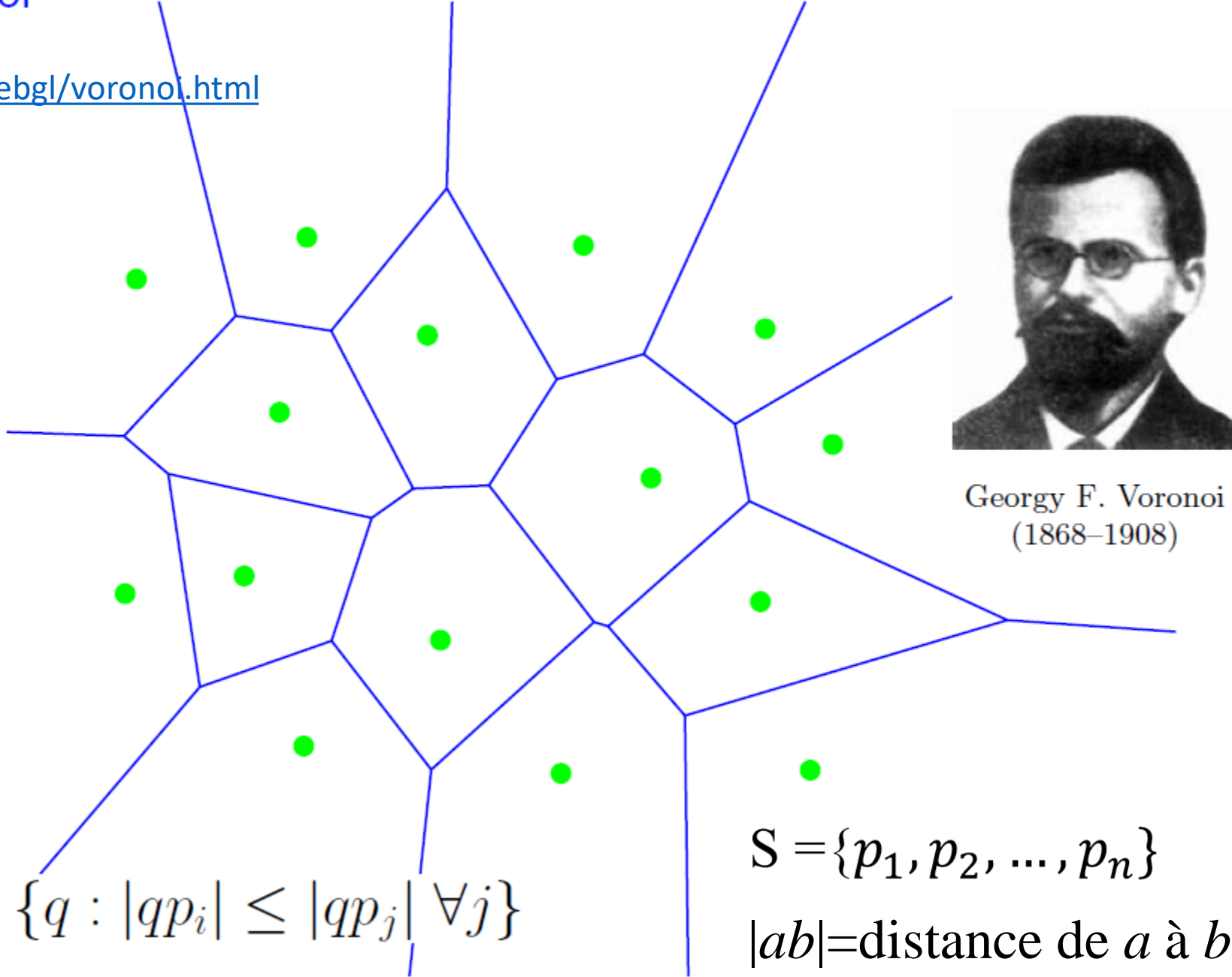


Voronoi is everywhere



# Voronoi

<http://alexbeutel.com/webgl/voronoi.html>



Georgy F. Voronoi  
(1868–1908)

$$V_i = \{q : |qp_i| \leq |qp_j| \forall j\}$$

$$S = \{p_1, p_2, \dots, p_n\}$$

$|ab|$  = distance de  $a$  à  $b$



# Diagramme de Voronoï

❖ Une région de Voronoï d'un  $p_i$  est définie par :

$$R(p_i, S) = \{p \in \mathbb{R}^2; |p_i p| \leq |p_j p| \forall j \neq i\}$$

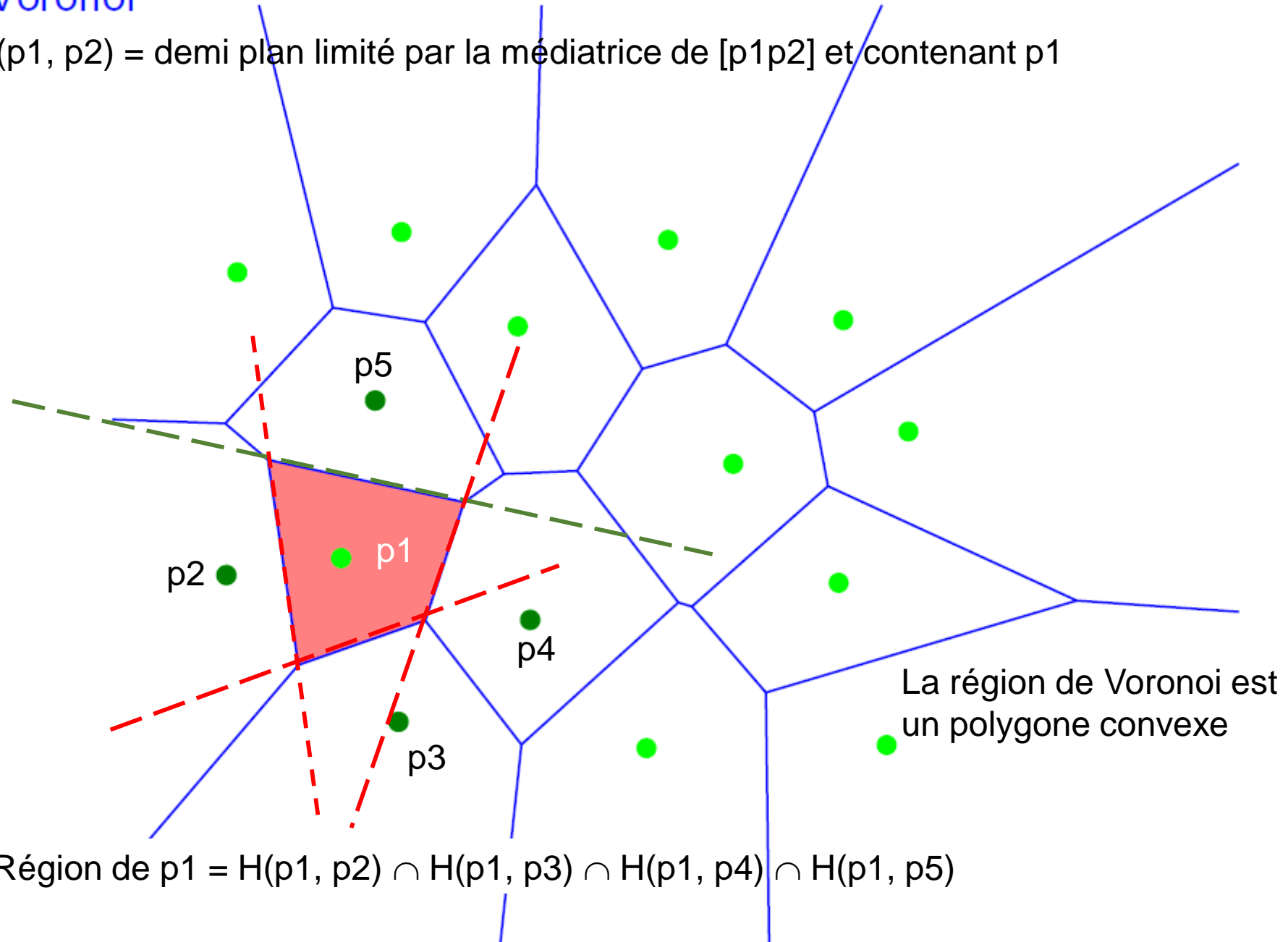
❖ Le diagramme de Voronoï de l'ensemble  $S$ ,  $DV(S)$ , est l'ensemble des régions  $R(p_i, S)$ .

$$\text{Propriété : } R(p_i, S) = \bigcap_{i \neq j} H(p_i, p_j)$$

$H(p_i, p_j)$  = demi plan limité par la médiatrice de  $[p_i p_j]$  et contenant  $p_i$ .

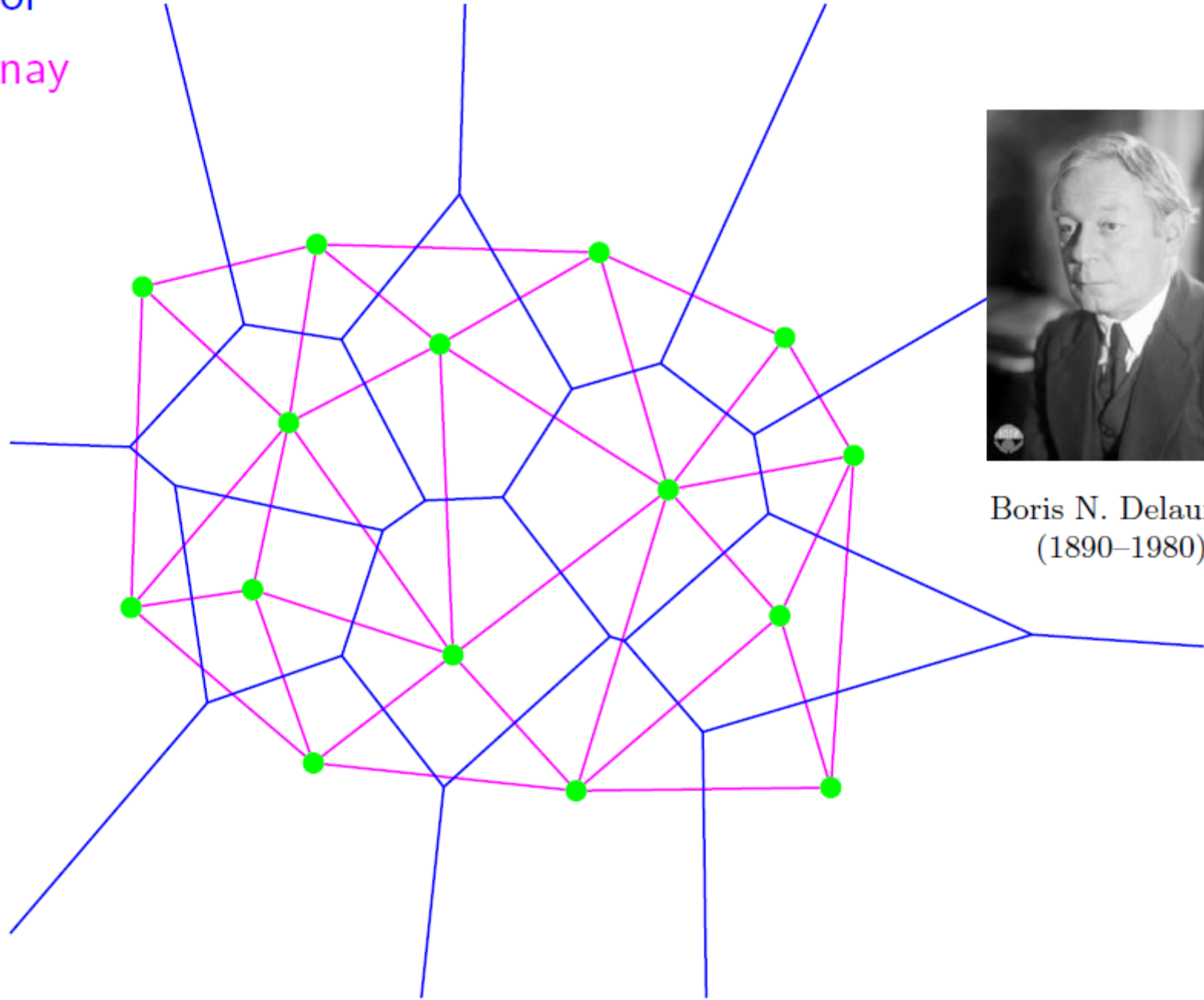
# Voronoi

$H(p_1, p_2)$  = demi plan limité par la médiatrice de  $[p_1p_2]$  et contenant  $p_1$



Région de  $p_1 = H(p_1, p_2) \cap H(p_1, p_3) \cap H(p_1, p_4) \cap H(p_1, p_5)$

Voronoi  
Delaunay



Boris N. Delaunay  
(1890–1980)

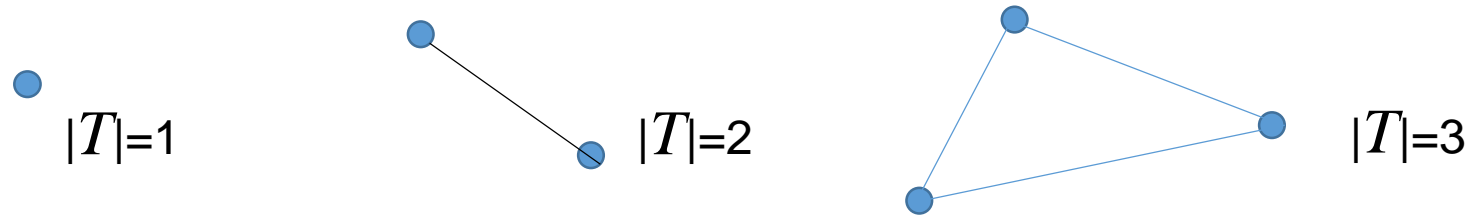


# Triangulation de Delaunay

$S = \{p_1, p_2, p_3, \dots, p_n\}$  est un ensemble fini de points sur un plan Euclidien.

Soient  $T \subset S$ , tel que  $|T| \leq 3$ , et  $\sigma_T = \text{Conv}(T)$

$|T|$  = nombre de points dans  $T$ .  $\text{Conv}(T)$  = enveloppe convexe de  $T$ .

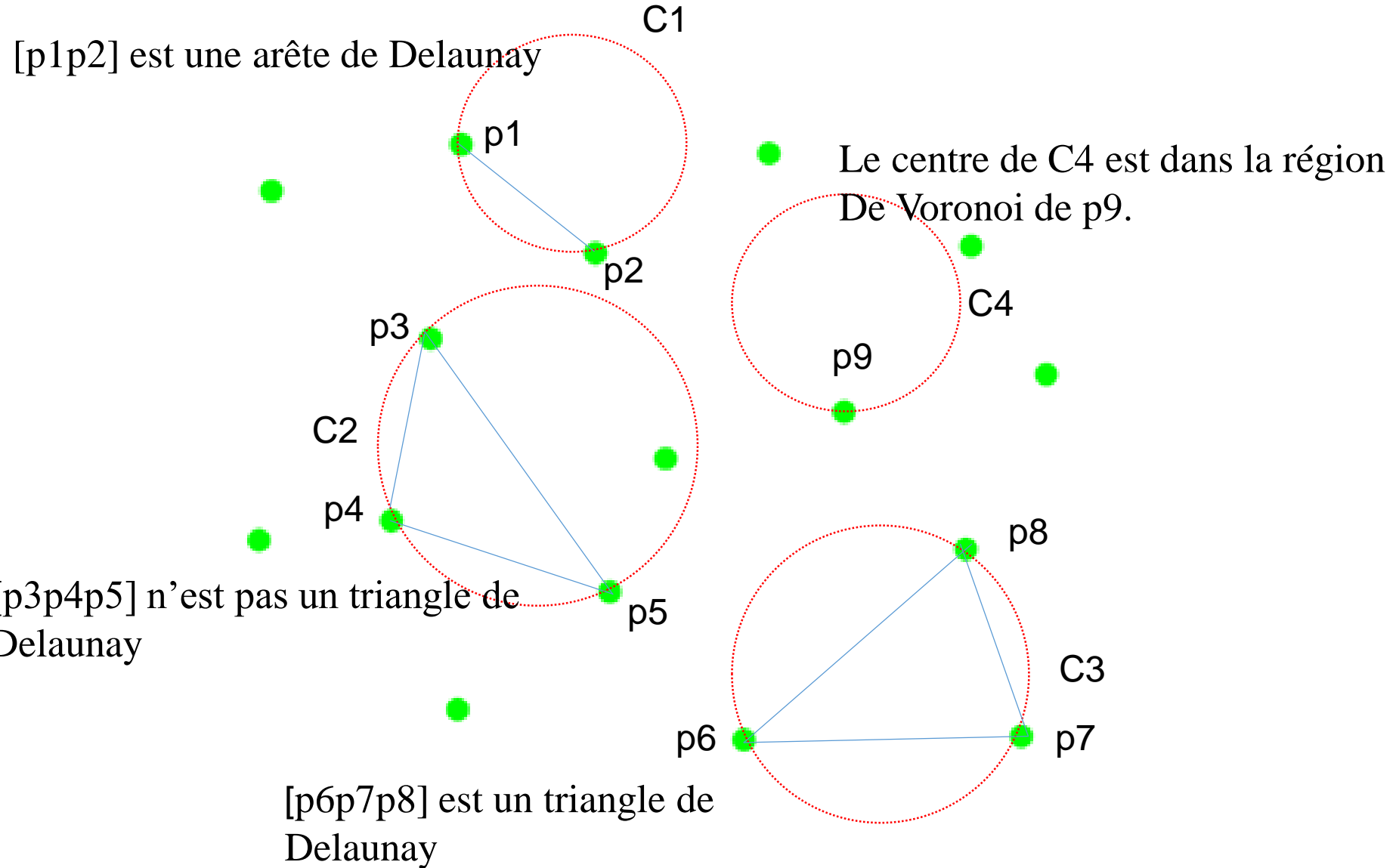


$\sigma_T$  est un point, un segment, ou un triangle.

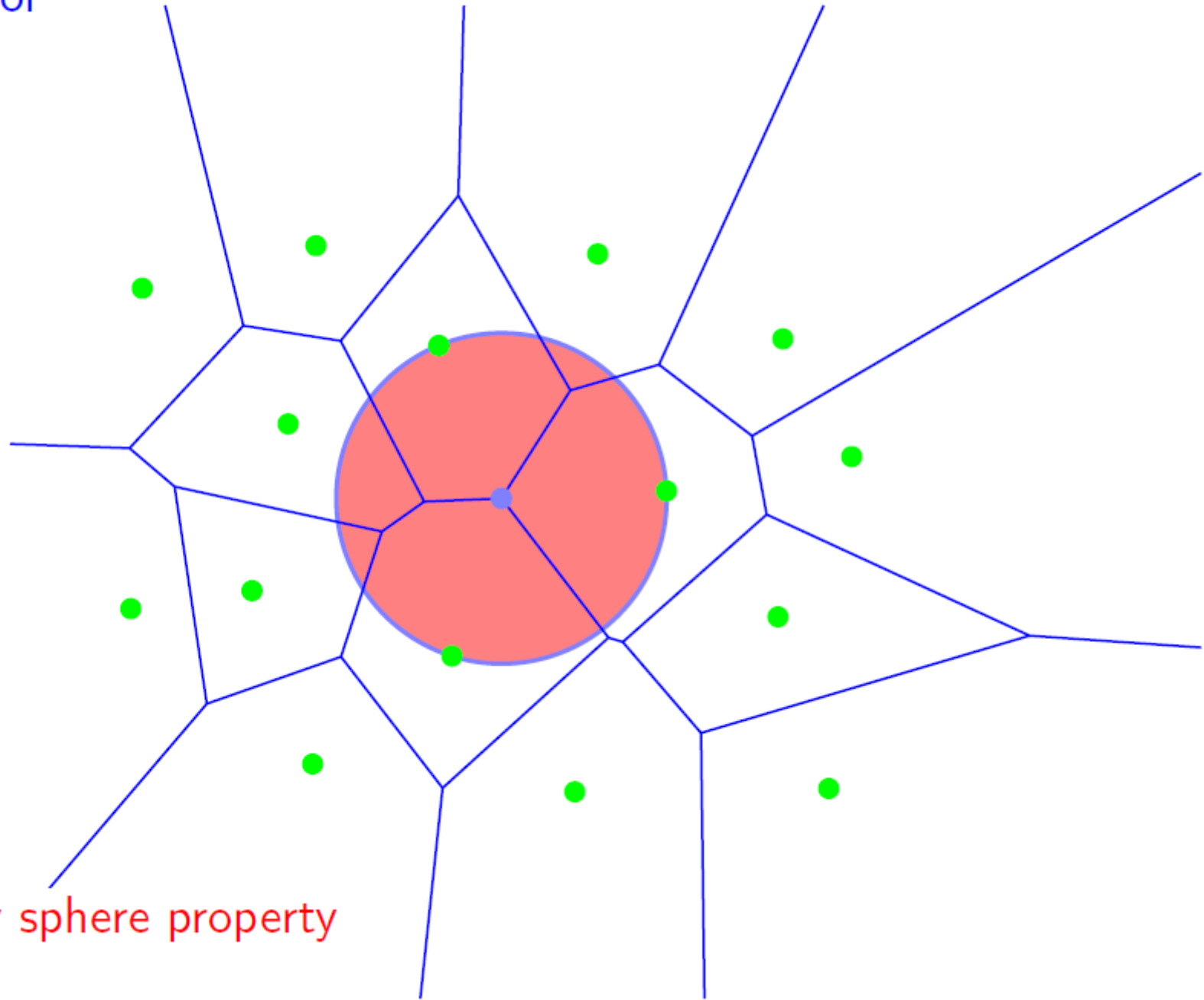
$\sigma_T = \text{Conv}(T)$  appartient à la triangulation de Delaunay de  $S$  si il existe un disque ouvert  $b$  tel que :

$$b \cap S = \emptyset \text{ et } T \subset \partial b \cap S$$

# Delaunay



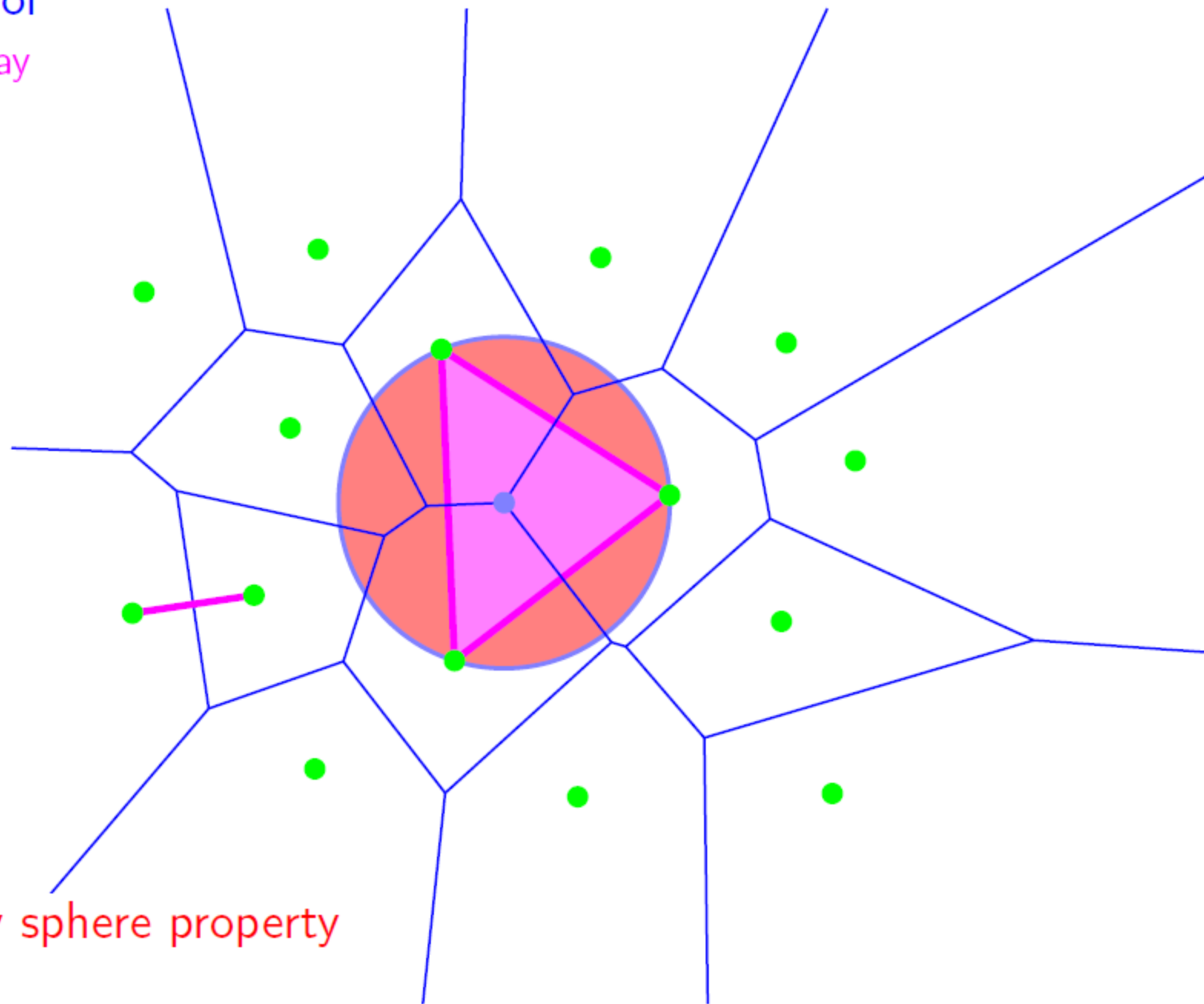
Voronoi



Empty sphere property

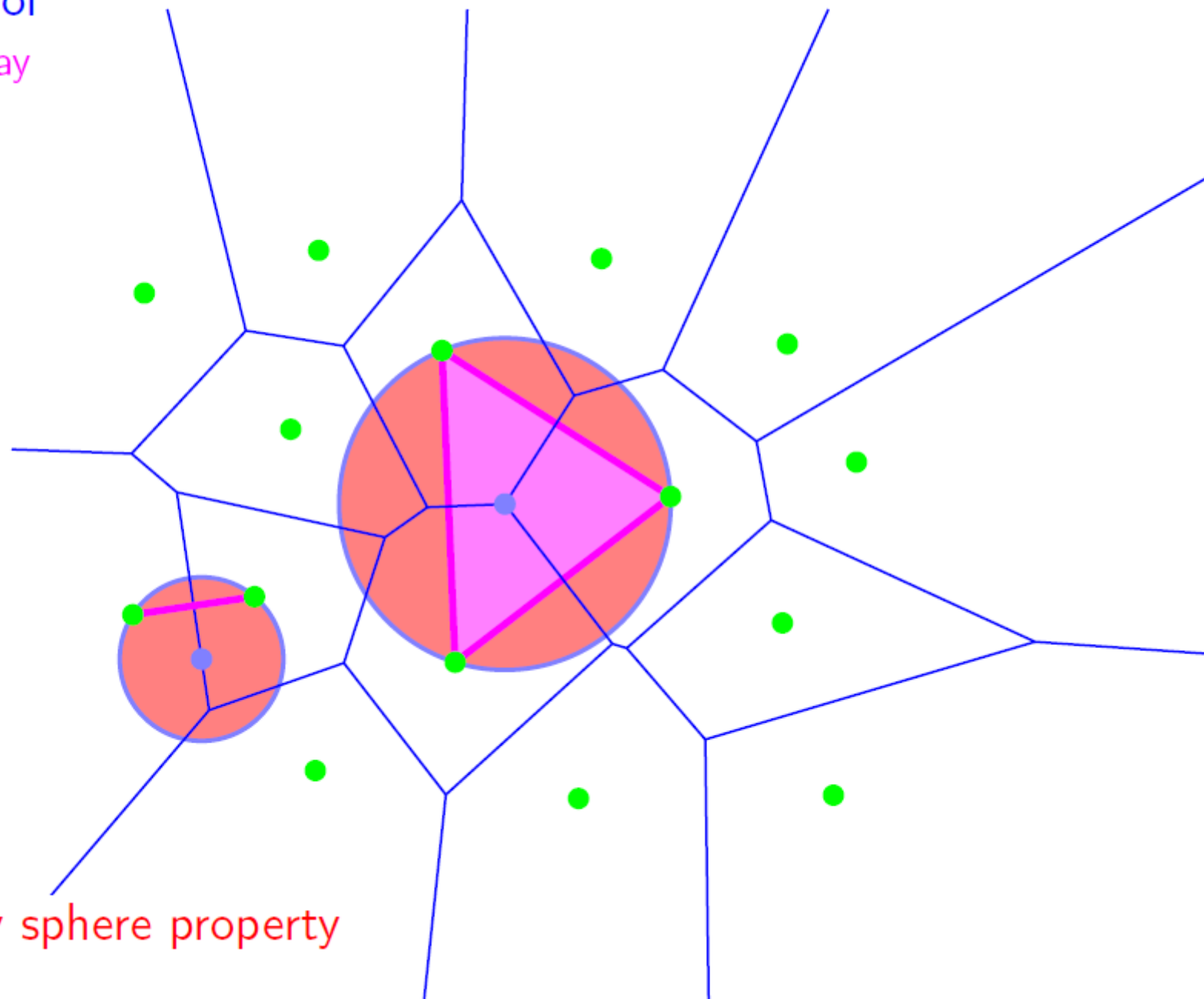


Voronoi  
Delaunay



Empty sphere property

Voronoi  
Delaunay



## 2. Diagramme de Voronoï discret

### 2.1. Algorithme de force brute

Etant donnés une image  $I$  de taille  $N \times M$  et un ensemble de pixels  $S = \{p_1, p_2, \dots, p_n\}$ . Posons  $I(p_i) = i$ .

Pour  $x = 0$  à  $N-1$  faire

Pour  $y = 0$  à  $M-1$  faire {

$P = (x, y)$ ;

Calculer l'indice  $m$  tel que  $d(P, p_m) \leq d(P, p_i) \forall i = 1 \dots n$

$I(x, y) = m.$ }

$P = (x, y), P' = (x', y')$

$$d_e(P, P') = \sqrt{(x - x')^2 + (y - y')^2}$$

$$d_1(P, P') = |x - x'| + |y - y'|$$

$$d_\infty(P, P') = \max(|x - x'|, |y - y'|)$$

Tester cet algorithme avec ces différentes distances



## 2. Diagramme de Voronoï discret

### 2.2. Algorithme séquentiel

On calcule la transformation de distances (TD) sur une image initial (0, infini) en balayant l'image deux fois de la façon suivante:

- avec le masque avant: de gauche à droite et de haut en bas,
- avec le masque arrière: de droite à gauche et de bas en haut.

# Algorithme séquentiel

C2 C1 C2

C1 0

Masque 3x3 avant

0 C1

C2 C1 C2

Masque 3x3 arrière

## Exemples de masques

1	1	1	4	3	4	$\sqrt{2}$	1	$\sqrt{2}$
1	0	1	3	0	3	1	0	1
1	1	1	4	3	4	$\sqrt{2}$	1	$\sqrt{2}$

$\sqrt{2}$  1  $\sqrt{2}$

1 0

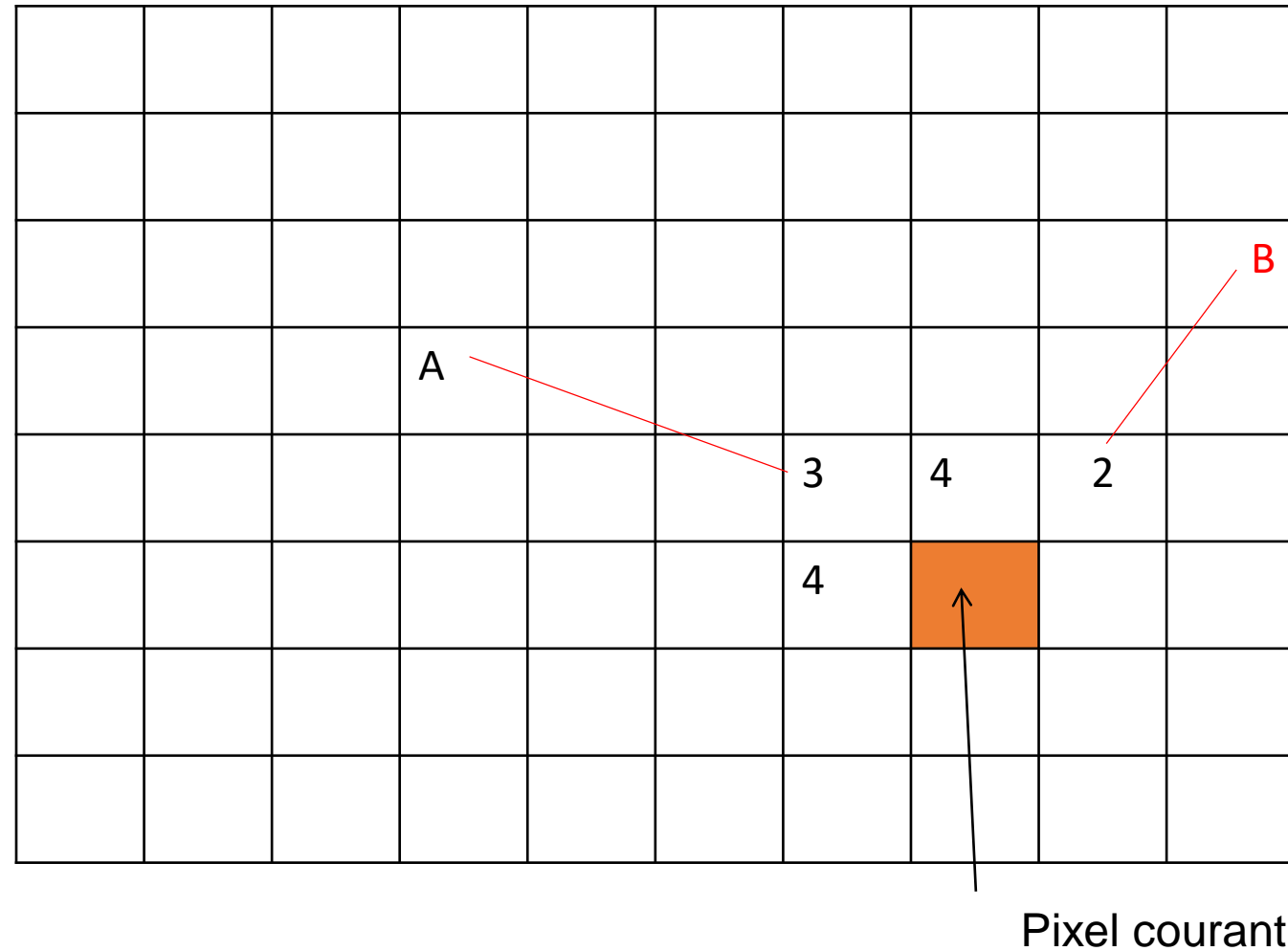
Masque 3x3 avant

0 1

$\sqrt{2}$  1  $\sqrt{2}$

Masque 3x3 arrière

# Algorithme séquentiel





# Algorithme séquentiel

$D_{i,j}=0$  si  $(i,j) \in \text{Objet}$

$D_{i,j} = \infty$  si  $(i,j) \notin \text{Objet}$

**Etape avant :**                      taille = 3 pour un masque 3x3

Pour  $i = (\text{taille}+1)/2$  à nblignes faire

    Pour  $j = (\text{taille}+1)/2$  à nbcolonnes faire

$$D_{i,j} = \min_{(k,l) \in \text{masque avant}} (D_{i+k,j+l} + C(k,l))$$

**Etape arrière :**

Pour  $i = \text{nblignes} - (\text{taille}+1)/2$  à 1 faire

    Pour  $j = \text{colonnes} - (\text{taille}+1)/2$  à 1 faire

$$D_{i,j} = \min_{(k,l) \in \text{masque arrière}} (D_{i+k,j+l} + C(k,l))$$

# Algorithme séquentiel $S = \{P_1, P_2, \dots, P_n\}$

$$\begin{aligned} D_{i,j} &= 0 & \text{si } (i, j) &= P_m & V_{i,j} &= m & \text{si } (i, j) &= P_m \\ D_{i,j} &= \infty & \text{si } (i, j) &\neq P_m & V_{i,j} &= 0 & \text{si } (i, j) &\neq P_m \end{aligned}$$

Etape avant :

(taille = 3)

Pour  $i = (\text{taille}+1)/2$  à nblignes faire

Pour  $j = (\text{taille}+1)/2$  à nbcolonnes faire

$$\begin{aligned} D_{i,j} &= D_{i+k, j+l} + C(k, l) = \min(D_{i-1, j-1} + C(-1, -1), D_{i-1, j} + C(-1, 0), \\ &D_{i-1, j+1} + C(-1, 1), D_{i, j-1} + C(0, -1), D_{i, j} + C(0, 0)) \end{aligned}$$

$$V_{i,j} = V_{i+k, j+l}$$

Etape arrière :

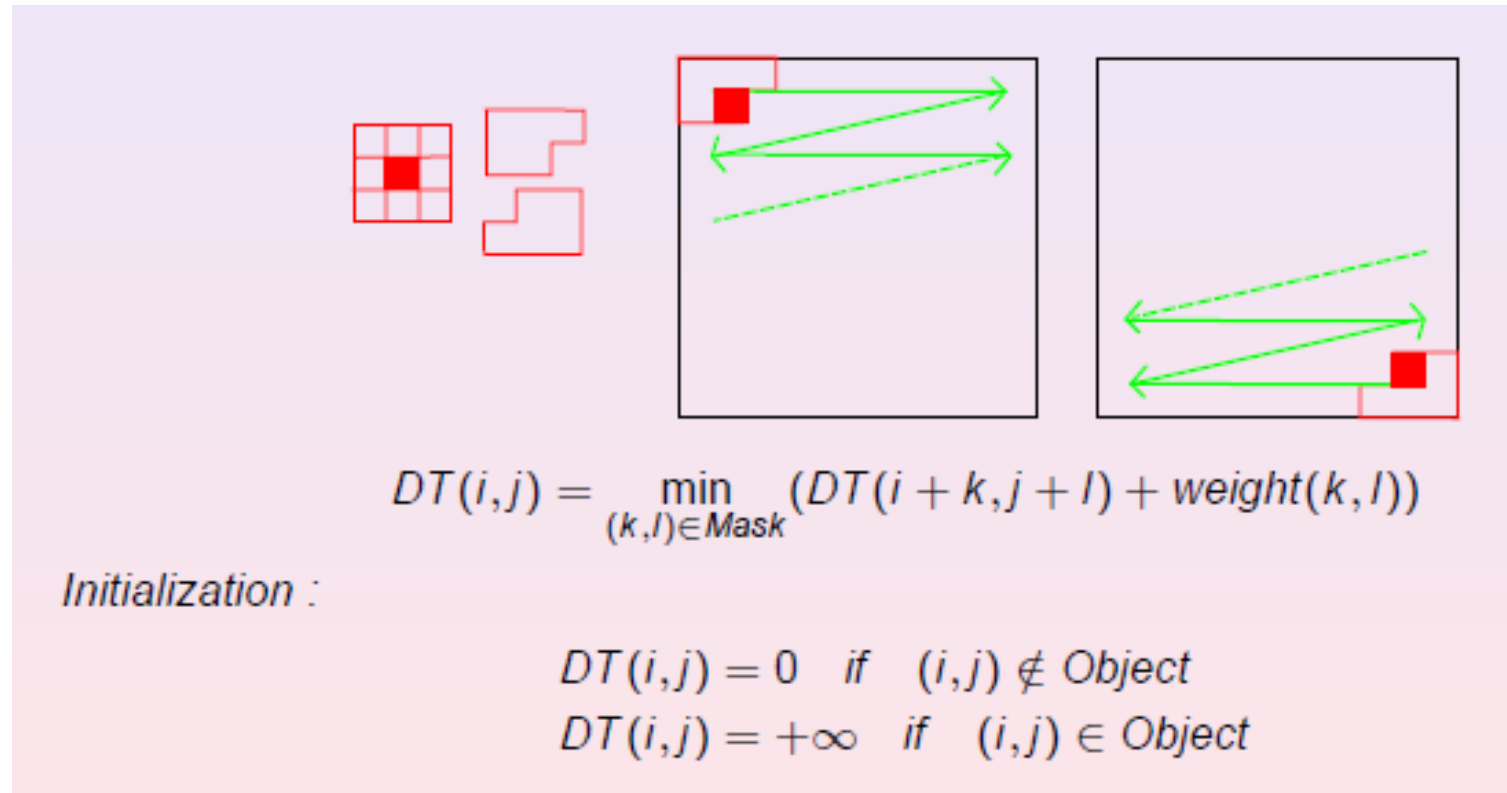
Pour  $i = \text{nblignes} - (\text{taille}+1)/2$  à 1 faire

Pour  $j = \text{colonnes} - (\text{taille}+1)/2$  à 1 faire

$$\begin{aligned} D_{i,j} &= D_{i+k, j+l} + C(k, l) = \min(D_{i, j+1} + C(0, 1), D_{i+1, j+1} + C(1, 1), \\ &D_{i+1, j} + C(1, 0), D_{i+1, j-1} + C(1, -1), D_{i, j} + C(0, 0)) \end{aligned}$$

$$V_{i,j} = V_{i+K, j+L}$$

# Algorithme séquentiel





## Exemple (distance $d_\infty$ )

### Distance

-	-	-	-	-	-	-
-	-	-	-	-	-	-
-	<b>0</b>	-	-	-	-	-
-	-	-	<b>0</b>	-	-	-
-	-	-	-	-	-	-
-	-	-	-	<b>0</b>	-	-
-	-	-	-	-	-	-

-	-	-	-	-	-	-
-	-	-	-	-	-	-
-	<b>0</b>	1	2	3	4	5
1	1	1	<b>0</b>	1	2	3
2	2	1	1	1	2	3
3	2	2	2	0	1	2
3	3	3	1	1	1	2

### Voronoi

-	-	-	-	-	-	-
-	-	-	-	-	-	-
-	<b>1</b>	-	-	-	-	-
-	-	-	<b>2</b>	-	-	-
-	-	-	-	-	-	-
-	-	-	-	<b>3</b>	-	-
-	-	-	-	-	-	-

-	-	-	-	-	-	-
-	-	-	-	-	-	-
-	<b>1</b>	1	1	1	1	1
1	1	1	<b>2</b>	2	2	2
1	1	2	2	2	2	2
1	2	2	2	<b>3</b>	3	3
2	2	2	3	3	3	3

## Exemple

### Distance

-	-	-	-	-	-	-
-	-	-	-	-	-	-
-	<b>0</b>	1	2	3	4	5
1	1	1	<b>0</b>	1	2	3
2	2	1	1	1	2	3
3	2	2	2	0	1	2
3	3	3	1	1	1	2

2	2	2	2	3	3	3
1	1	1	2	2	2	3
1	<b>0</b>	1	1	1	2	3
1	1	1	<b>0</b>	1	2	2
2	2	1	1	1	1	2
3	2	2	1	0	1	2
3	3	2	1	1	1	2

### Voronoi

-	-	-	-	-	-	-
-	-	-	-	-	-	-
-	<u>1</u>	1	1	1	1	1
1	1	1	<u>2</u>	2	2	2
1	1	2	2	2	2	2
1	2	2	2	<u>3</u>	3	3
2	2	2	3	3	3	3

1	1	1	1	1	2	2
1	1	1	1	2	2	2
1	<u>1</u>	1	2	2	2	2
1	1	1	<u>2</u>	2	2	3
1	1	2	2	2	3	3
1	2	2	3	<u>3</u>	3	3
2	2	3	3	3	3	3

# Algorithme parallèle

À l'itération  $m$  :

$$V_{i,j}^m = \min_{(k,l) \in \text{masque}} (V_{i+k,j+l}^{m-1} + C(k,l))$$

C2 C1 C2

C1 0 C1

C2 C1 C2

Masque distance

# Exemple

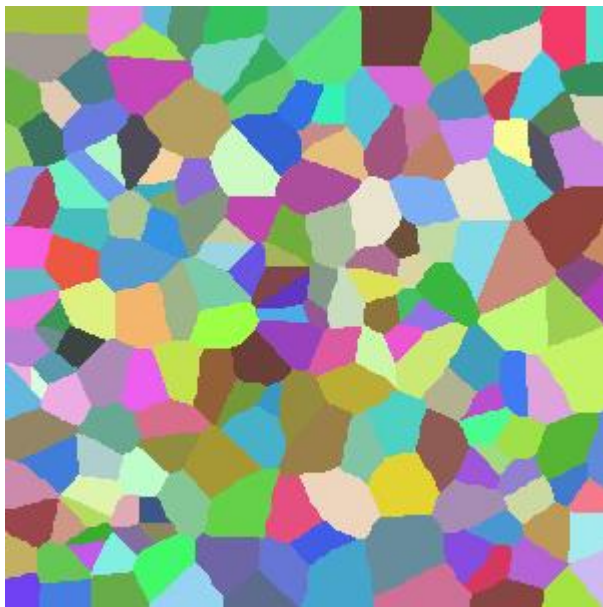
-	-	-	-	-	-	-
-	-	-	-	-	-	-
-	-	1	1	1	-	-
-	-	1	0	1	-	-
-	-	1	1	1	-	-
-	-	-	-	-	-	-
-	-	-	-	-	-	-

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

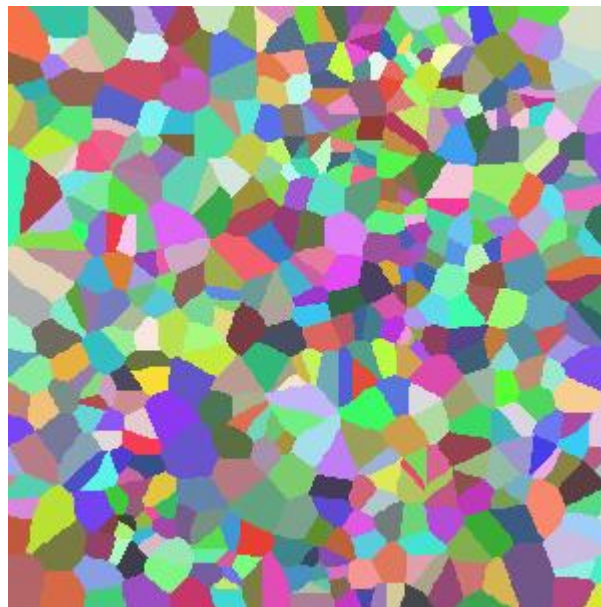
3	3	3	3	3	3	3
3	2	2	2	2	2	3
3	2	1	1	1	2	3
3	2	1	0	1	2	3
3	2	1	1	1	2	3
3	2	2	2	2	2	3
3	3	3	3	3	3	3



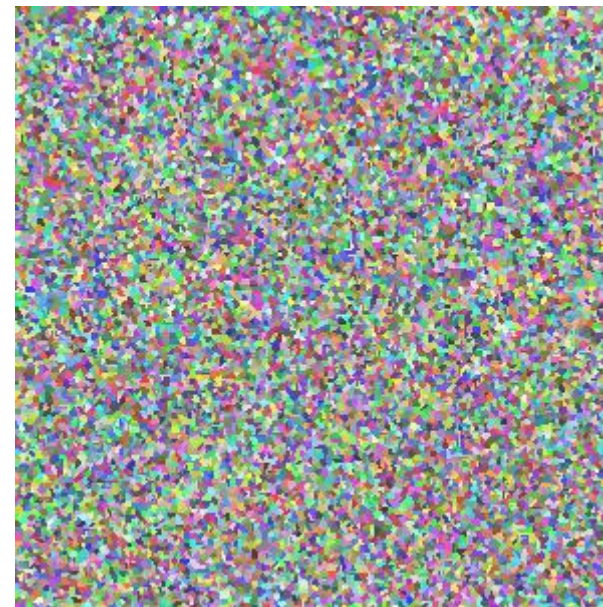
# Exemples



200 points



500 points



1500 points

### 3. Approximation d'images en utilisant le diagramme de Voronoï



## 3.1. Approximation d'images en utilisant le diagramme de Voronoï

### Algorithme de force brute

Considérons une image  $I$  de taille  $N=M \times M$

- 1) Générer aléatoirement un grand nombre de points (germes)  $n$ , sur le support image.
- 2) Calculer le diagramme de Voronoï en utilisant les germes sélectionnés.
- 3) Associer aux pixels de chaque région la moyenne des niveaux de gris.

**TP3.** Mettre en œuvre cet algorithme

## 3.2. Approximation d'images en utilisant le diagramme de Voronoï

### Algorithme adaptatif

Considérons une image  $I$  de taille  $N=M \times M$

- 1) Générer aléatoirement un nombre faible de points  $n$ , sur le support image. (e.g. 0.2% de l'ensemble de pixels ;  $n=2N/1000$ ).
- 2) Calculer le diagramme de Voronoï en utilisant les germes sélectionnés.
- 3) Associer aux pixels de chaque région la moyenne des niveaux de gris.
- 4) Pour chaque région de Voronoï, calculer la variance  $V$  et identifier les régions homogènes. Une région est homogène ssi  $V < \text{seuil}$ .

On peut aussi utiliser le coefficient de variation ( $CF = \text{écart-type}/\text{moyenne}$ ).



# Approximation d'images en utilisant le diagramme de Voronoi (suite)

- 5) Soit  $T$  le nombre de pixels d'une région  $R$  non-homogène. Sélectionner uniformément dans  $R$  un nombre d'échantillons  $m = 10\% * T$ . Ajouter ces échantillons aux points précédemment sélectionnés
- 6) **Réitérer** les étapes 2) à 5) **jusqu'à** ce que le nombre des nouveaux échantillons soit 0.

**TP4.** Mettre en œuvre cet algorithme

## 4. Algorithme K-moyennes (K-means)

**Entrée** : un ensemble de points  $S$ ,  $k$  (nombre de classes)

**Sortie** : Les classes  $\mathbf{C}_k$

Choisir  $k$  points  $c_1 \dots c_k$  ( $k$  centres de classes  $\mathbf{C}_k$ ),

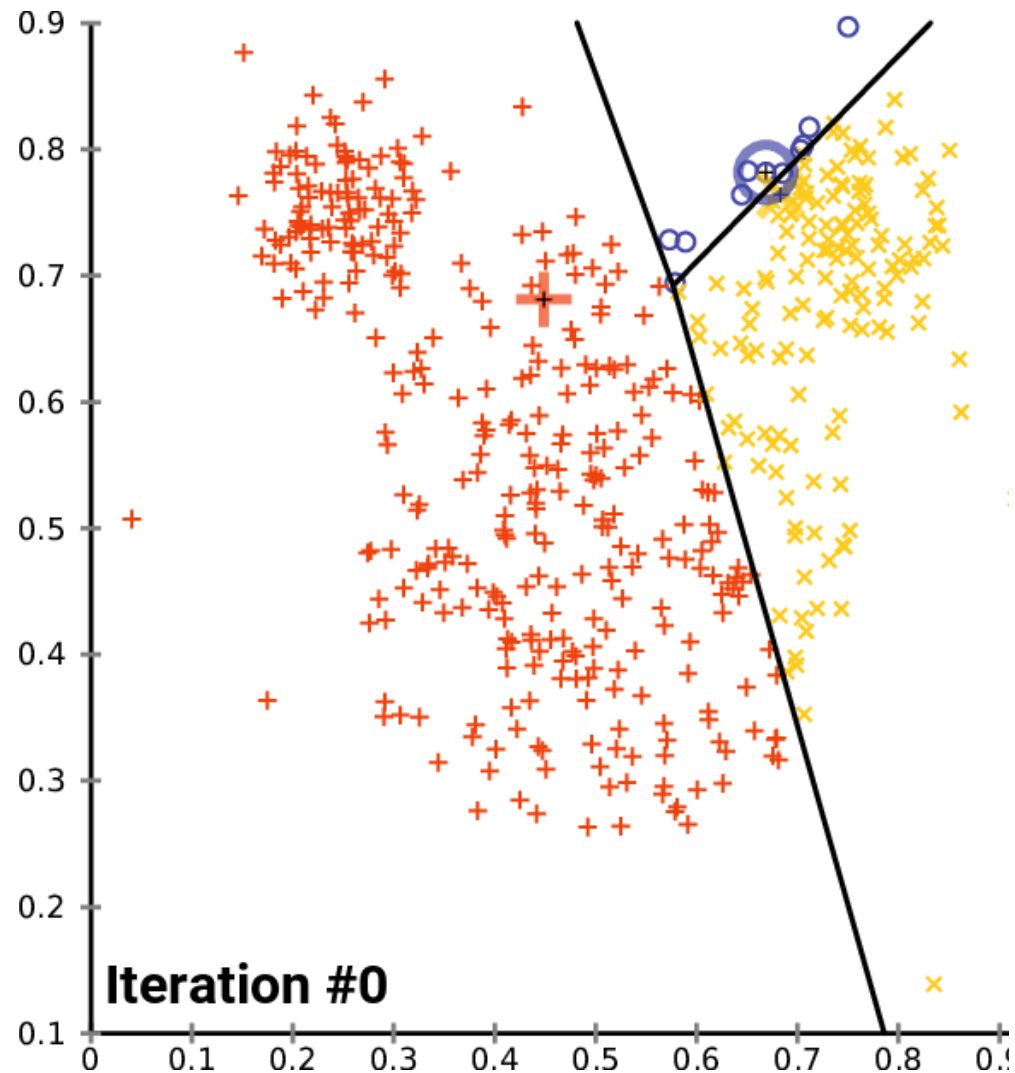
**Répéter**

- Calculer le diagramme de Voronoï des centres  $c_1 \dots c_k$
- Affecter  $p_j \in S$  à la classe  $\mathbf{C}_i$  si  $p_j \in R(c_i)$ .  
(affecter l'étiquette  $i$  à l'individu  $p_j$ )
- Calculer le nouveau centre de gravité  $c_i$  de la classe  $\mathbf{C}_i$ .

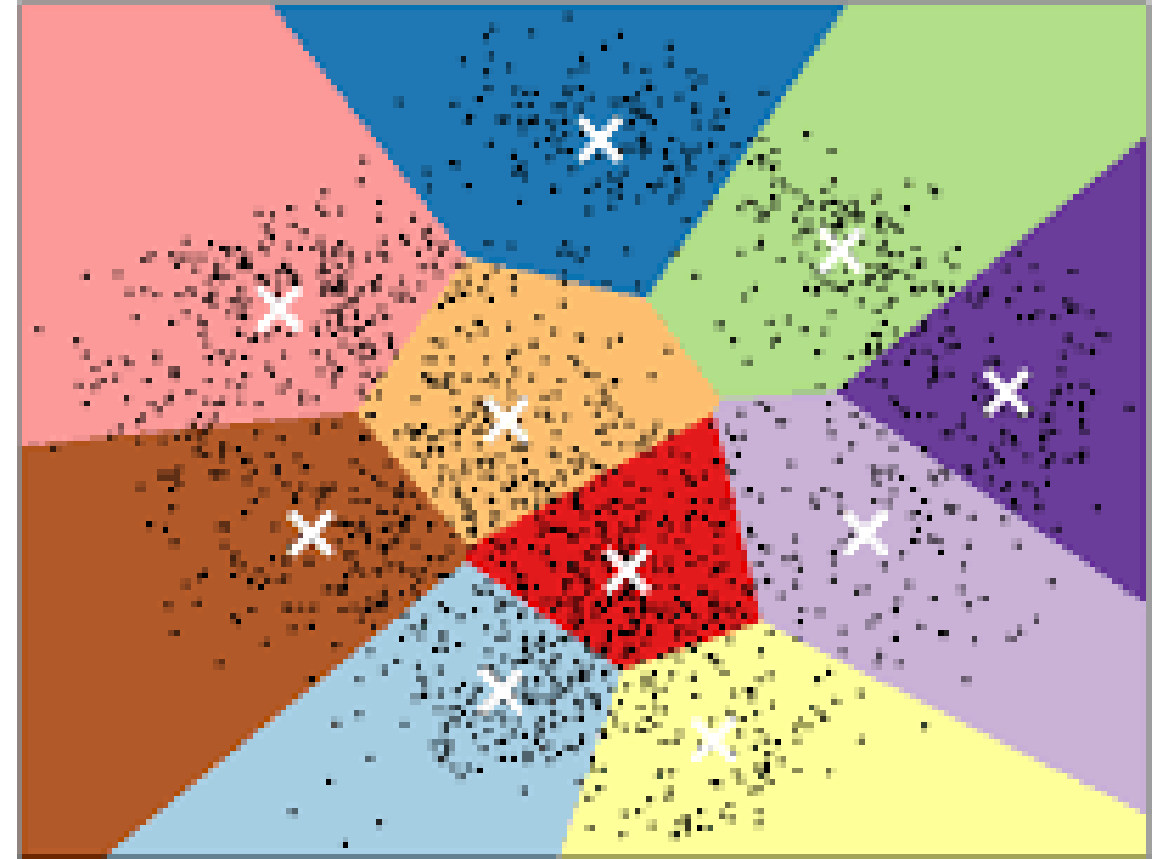
**Jusqu'à** ( $c_i$  ne change pas ou le nombre d'itération dépasse NMAX)

**TP4.** Mettre en œuvre l'algorithme K-moyennes

# Algorithme K-moyennes (K-means)



K-means clustering on the digits dataset (PCA-reduced data)  
Centroids are marked with white cross



# Mise en œuvre de l'algorithme k-moyenne++

## Entrée :

- Données :  $X = \{x_1, x_2, \dots, x_n\}$ , - Nombre de clusters  $k$

## Sortie :

- Centres des clusters  $C = \{C_1, C_2, \dots, C_k\}$
- - Assignment des points aux clusters

### 1. Initialisation des centres :

- 1.1. Choisir aléatoirement un centre  $C_1$  parmi les points de  $X$ .
- 1.2. Pour chaque point  $x_i \in X$ , calculer la distance au centre le plus proche :

$$D(x_i) = \min_{j=1}^k |x_i - C_j|^2$$

- 1.3 Choisir un nouveau centre  $C_2$  avec une probabilité proportionnelle  $D(x_i)^2$
- 1.4 Répéter les étapes 1.2 et 1.3 jusqu'à avoir  $k$  centres  $C = \{C_1, C_2, \dots, C_k\}$ .

### 2. Répéter jusqu'à convergence :

- 2.1 Pour chaque point  $x_i \in X$ , assigner  $x_i$  au centre le plus proche  $C_K$ :

$$|x_i - C_K|^2 = \min_{j=1}^n (|x_i - C_j|^2)$$

(i.e Soient  $V(C_1) \dots V(C_n)$  les régions de Voronoi, trouver  $C_k$  tel que  $x_i \in V(C_K)$ )

- 2.2 Pour chaque cluster, recalculer le centre comme la moyenne des points :

$$C_k = \frac{\sum_{x_i \in \text{cluster}(C_k)} x_i}{\text{card}(C_k)}$$

- 2.3 Vérifier la convergence : Si les centres ne changent plus ou changent très peu, arrêter.

Sortie : - Les centres  $C$  - Les clusters assignés pour chaque point.



## Comment sélectionner les premiers centres $C_j$ ?

Nous voulons sélectionner un point  $x_i$  comme nouveau centre  $C_j$ , avec une probabilité proportionnelle au carré de la distance  $D(x_i)^2$ , où  $D(x_i)$  est la distance minimale du point  $x_i$  au centre déjà sélectionné.

Étapes détaillées pour choisir la probabilité proportionnelle à  $D(x_j)^2, \dots, D(x_n)^2$  :

1. Calculer la somme des distances au carré :

Pour chaque point  $x_i$ , on a déjà calculé la distance  $D(x_i)$ . Ensuite, on calcule la somme des distances au carré :

$$S = \sum_{i=j}^n D(x_i)^2$$

2. Calculer la probabilité pour chaque point :

La probabilité  $P(x_i)$  de sélectionner le point  $x_i$  comme nouveau centre est proportionnelle à  $D(x_i)^2$ . Pour obtenir la probabilité exacte, on divise  $D(x_i)^2$  par la somme totale des distances au

carré  $S$ . La probabilité est donc donnée par :

$$P(x_i) = \frac{D(x_i)^2}{S}$$

où  $S = \sum_{k=j}^n D(x_k)^2$  est la somme des distances au carré pour tous les points  $x_2, x_3, \dots, x_n$ .

### 3. Tirer un point en utilisant cette probabilité :

Pour choisir le nouveau centre  $C_2$ , on peut appliquer une technique de tirage probabiliste (souvent appelée "roulette wheel selection"). Voici comment procéder :

- Générer un nombre aléatoire  $r$  dans l'intervalle  $[0, 1]$ .
- Construire les probabilités cumulées pour chaque point  $x_i$  :

$$P_{\text{cumul}}(x_i) = \sum_{k=j}^i P(x_k)$$

- Sélectionner le point  $x_i$  tel que la probabilité cumulée  $P_{\text{cumul}}(x_i)$  dépasse le nombre aléatoire  $r$ .

**Entrée :**

- Données :  $X[0] \dots X[n-1]$ ,  $k$  nombre de clusters

**Sortie :**

- Centres des clusters  $C[0] \dots C[k-1]$
- - Assignment des points aux clusters

**1. Initialisation des centres :**

1.1. Choisir aléatoirement un centre  $C[0]$  parmi les points de  $X$ .

1.2. Pour chaque point  $X[i]$ , calculer la distance au centre le plus proche :

$$D(i) = \min_{j=0}^{k-1} |X[i] - C[j]|^2$$

1.3 Choisir un nouveau centre  $C[1]$  avec une probabilité proportionnelle  $D(i)^2$

1.4 Répéter les étapes 1.2 et 1.3 jusqu'à avoir  $k$  centres  $C$ .

**2. Répéter jusqu'à convergence :**

2.1 Pour chaque point  $X[i]$ , assigner  $x_i$  au centre le plus proche  $C_K$ :

$$|X[i] - C[k]|^2 = \min_{j=1}^n (|X[i] - C[j]|^2)$$

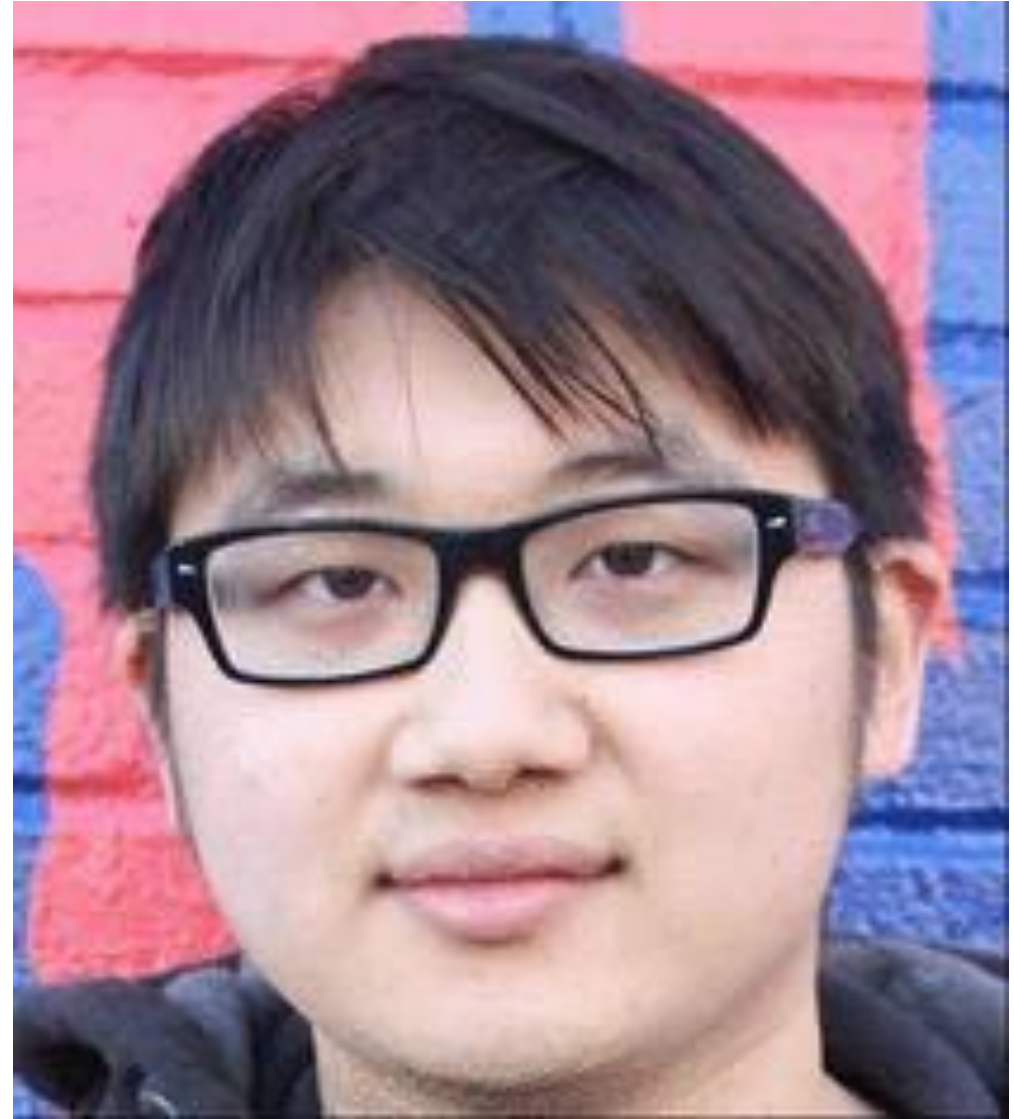
2.2 Pour chaque cluster, recalculer le centre comme la moyenne des points :

$$C_k = \frac{\sum_{X[i] \in \text{cluster}(C_k)} X[i]}{\text{card}(C_k)}$$

2.3 Vérifier la convergence : Si les centres ne changent plus ou changent très peu, arrêter.

Sortie : - Les centres  $C$  - Les clusters assignés pour chaque point.

## 4. Morphing d'images et triangulation de Delaunay



# Etapes de l'algorithme Morphing

L'objectif est de produire une séquence vidéo transformant une image M en une autre image N.

## Etape 1 : Marquer les points caractéristiques

Marquer les caractéristiques principales des images M et N, en établissant une correspondance un à un entre les points marqués dans M et N, (exemple mettre en correspondance deux visages, marquer les traits du visage dans les 2 images : les yeux, les sourcils, le nez, les lèvres, etc). Le nombre de points caractéristiques dans les deux images doit être le même.

Les caractéristiques peuvent être marquées manuellement ou en utilisant un logiciel pour détecter ces caractéristiques.



## Etape 2: Calculer les triangulations de Delaunay

Notons  $S_M = \{m_1, m_2, \dots, m_k\}$  et  $S_N = \{n_1, n_2, \dots, n_k\}$  les points caractéristiques de M et de N.

Soit  $\alpha$  une valeur réelle entre 0 et 1.

Calculer l'ensemble de points  $S_I = \{i_1, i_2, \dots, i_k\}$ , tel que

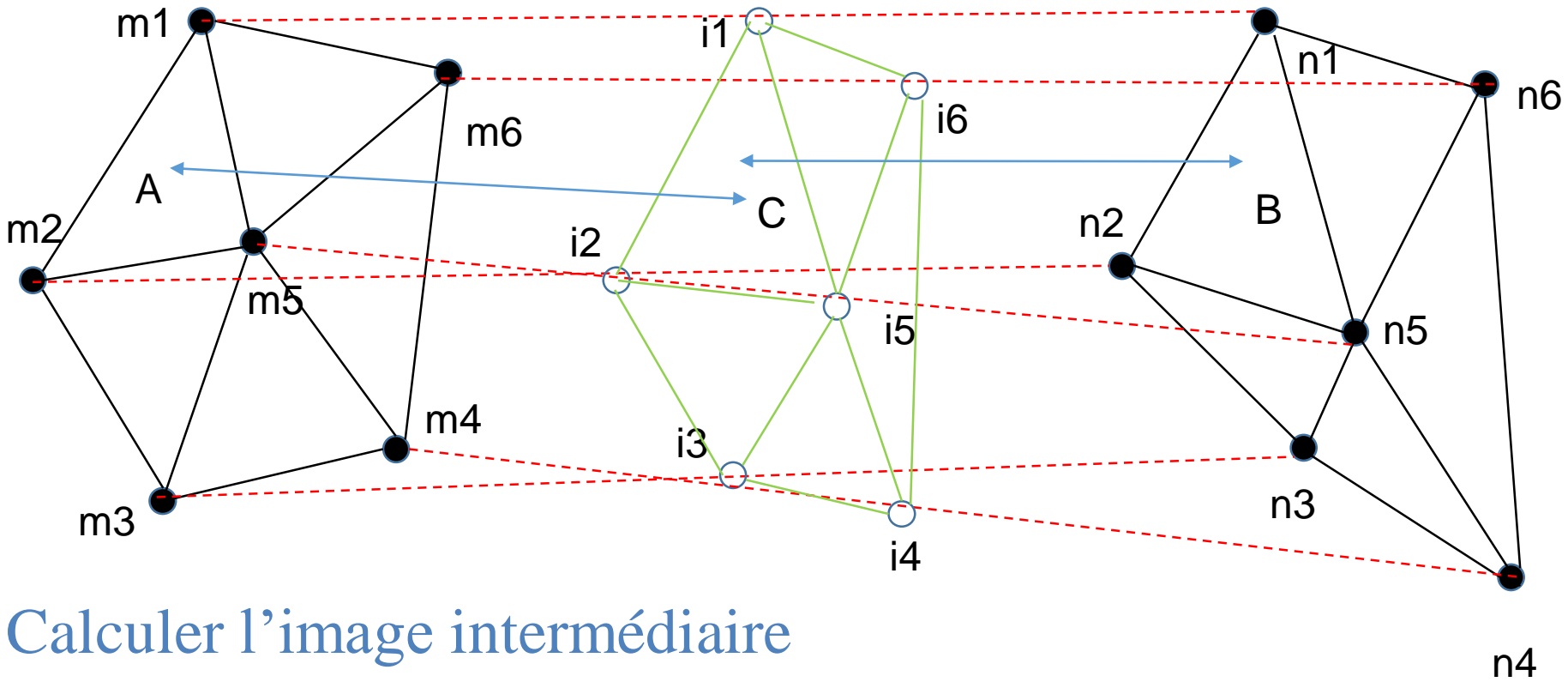
$$i_j = (1 - \alpha)m_j + \alpha n_j$$

Les points de  $S_I$  sont ceux de l'image intermédiaire.

Calculer les triangulations de Delaunay de  $S_M, S_N$  et  $S_I$ .

## Etape 2: Calculer les triangulations de Delaunay (suite)





## Etape 3 : Calculer l'image intermédiaire

### - Transformer le triangle A en B

Trouver la matrice T telle que  $TA=B$

$$\begin{pmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ 1 & 1 & 1 \end{pmatrix}$$

$$T = BA^{-1}$$

Chaque point  $(x,y)$  du triangle A est transformé en un point  $(x', y')$  de B en utilisant :

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = T \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- **Calculer les images M' et N'**

- Pour chaque triangle A de M et son correspondant C dans I, calculer la transformation  $T = AC^{-1}$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = T \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$M'(x', y') = M(x, y)$$

- Pour chaque triangle B de N et son correspondant C dans I, calculer la transformation  $T = BC^{-1}$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = T \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$N'(x', y') = N(x, y)$$

- Calculer l'image intermédiaire  $I(x, y) = (1 - \alpha)M'(x, y) + \alpha N'(x, y)$

La séquence vidéo est obtenue en variant  $\alpha$  de 0 à 1.

## - Fonctions d'OpenCv

Subdiv2D : calcule la triangulation de Delaunay et renvoie les listes des coordonnées des triangles.

getAffineTransform : calcule la matrice de transformation qui transforme un triangle en un autre.

## - Python: scipy.spatial :

<https://docs.scipy.org/doc/scipy/reference/tutorial/spatial.html>

## - Quelques liens

<https://devendrapratapyadav.github.io/FaceMorphing/>

<https://inst.eecs.berkeley.edu/~cs194-26/fa17/upload/files/proj4/cs194-26-aeh/>

<https://azmariewang.medium.com/face-morphing-a-step-by-step-tutorial-with-code-75a663cdc666>

<http://stackoverflow.com/questions/18844000/transfer-coordinates-from-one-triangle-to-another-triangle>

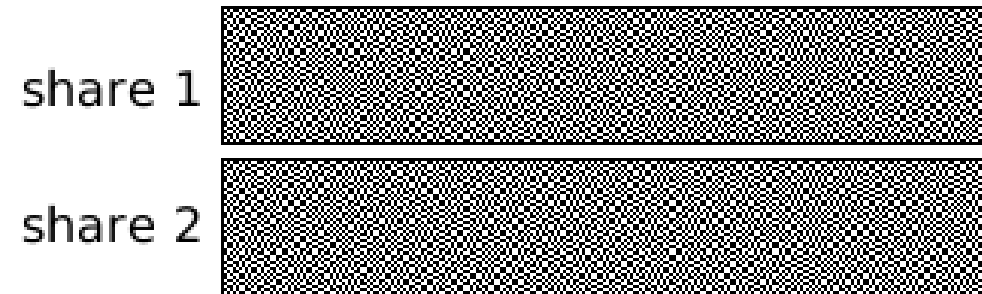
<https://www.learnopencv.com/delaunay-triangulation-and-voronoi-diagram-using-opencv-c-python/>

<https://www.cs.princeton.edu/courses/archive/fall00/cs426/papers/beier92.pdf>

<http://www.cs.cmu.edu/~quake/triangle.html>



## 6. Application à la cryptographie visuelle



## 6. Application à la cryptographie visuelle

Considérons une image binaire secrète  $S$  de taille  $N \times M$ .

Les pixels du message dans  $S$  ont la valeur 255 (blanc), les pixels du fond ont la valeur 0 (noir). L'image  $S$  est transformée en deux images clés  $K1$  et  $K2$  chacune de taille  $2N \times M$ .

Chacune des deux images est insuffisante pour déchiffrer l'image  $S$ .  
mais la superposition de  $K1$  et  $K2$  restitue le message de  $S$  (le XOR des deux donne l'image complète).

### Construction des 2 images clés



Masque 1



Masque 2

**Définition de  $K1$ .** Chaque pixel de l'image  $S$ , génère dans  $K1$  selon un processus aléatoire uniforme deux pixels horizontaux de type masque 1 ou masque 2.

**Définition de K2.** Si le pixel de S est blanc alors le masque généré dans K2 est le même que celui généré dans K1.

Si le pixel de S est noir alors le masque généré dans K2 est l'opposé de celui généré dans K1.















Pixel (if)	White 		Black 	
Probability	50%	50%	50%	50%
Mask for image key 1 (then)				
Mask for image key 2 (and)				
Resulting mask once image keys superposed				



Image secrète

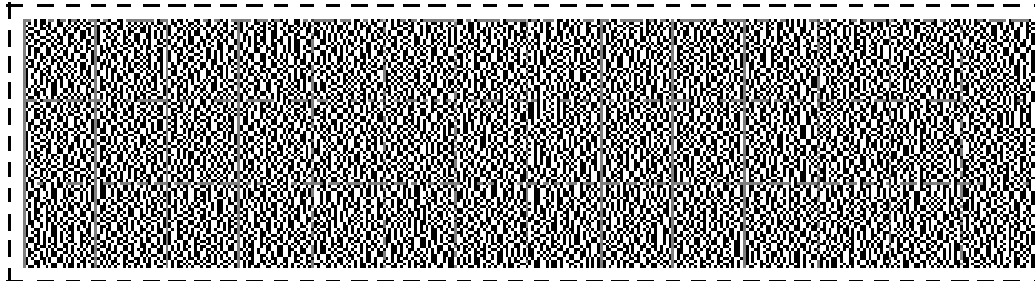


Image clé 1

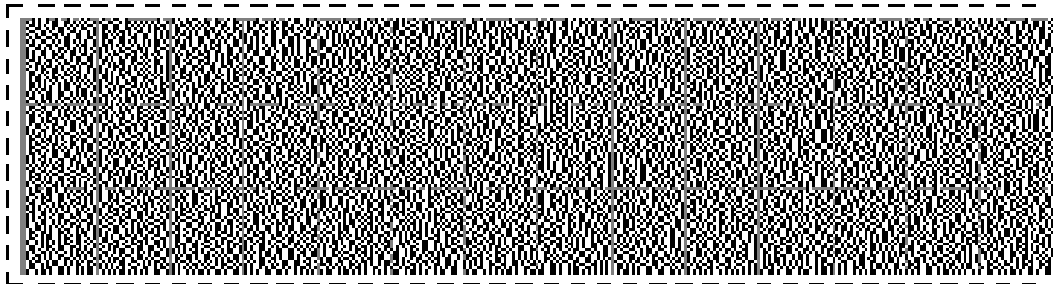


Image clé 2



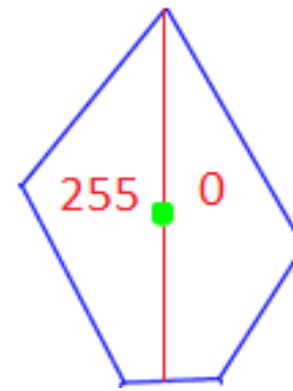
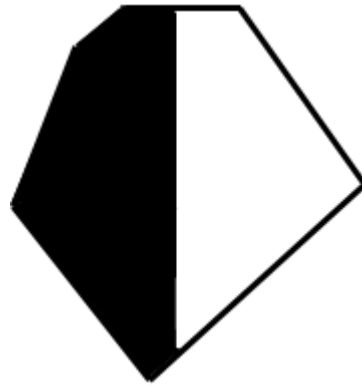
Image obtenue  
par la superposition  
des 2 images clés

# Cryptographie visuelle en utilisant le diagramme de Voronoï

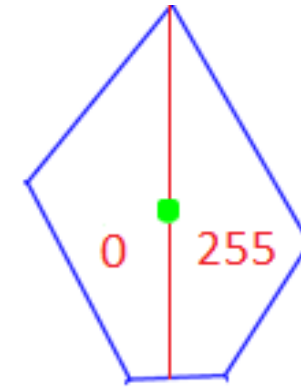
Approximation de l'image binaire secrète  $S$  de taille  $N \times M$  en utilisant le diagramme de Voronoï.

- ❑ Générer uniformément et aléatoirement  $n$  points sur la surface de  $S$ .
- ❑ Calculer le diagramme de Voronoï discret des points.
- ❑ Pour chaque région de Voronoï, calculer la proportion  $p_1$  des pixels blancs et la proportion  $p_2$  des pixels noirs.  
Si  $p_2 > p_1$  alors colorer les pixels de la région en noir sinon colorer en blanc la région de Voronoï.
- ❑ Choisir « le meilleur » nombre  $n$  pour obtenir une bonne approximation de l'image  $S$ .

# Cryptographie visuelle en utilisant le diagramme de Voronoï



Masque 1

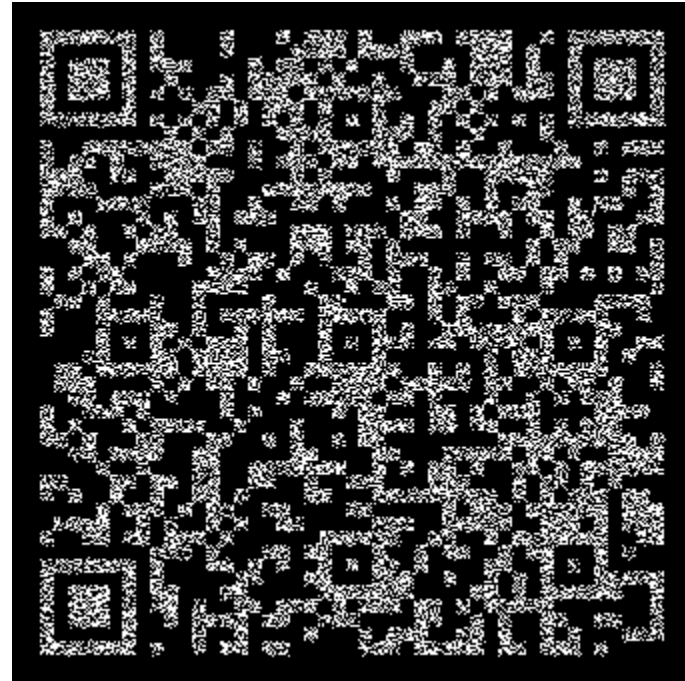
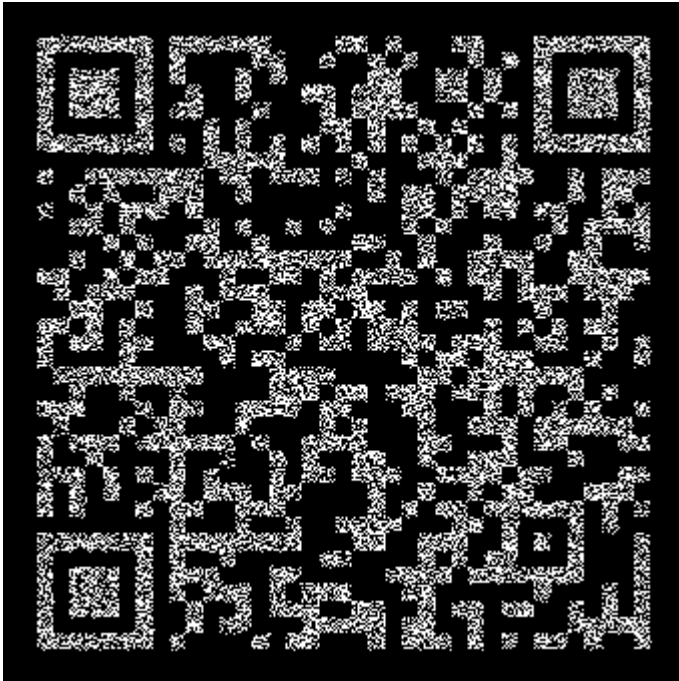


Masque 2

## Définition des masques.

- ❑ Calculer le centre de gravité de la région de Voronoï. Notons D la droite verticale passant pas le centre de gravité.
- ❑ Le masque 1 est une région de Voronoï telle que les pixels à gauche de la droite D sont en blanc et les pixels à droite sont en noir.
- ❑ Le masque 2 est obtenue en inversant les couleurs du masque 1.





**ZJ217**  
**X568E**

Image originale

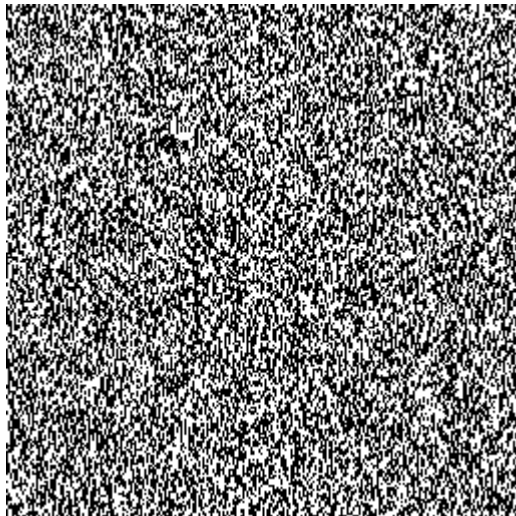
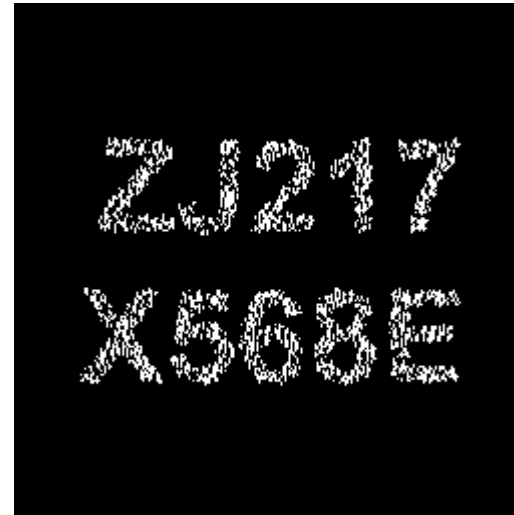


Image clé K1



Résultat de la superposition  
de K1 et K2

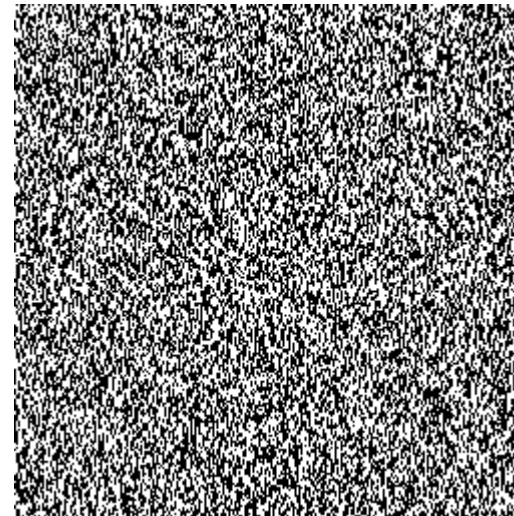


Image clé K2

20000 points