

# Réseaux géométriques pour l'indexation et la recherche de données en grandes dimensions

Loïc Paulevé

IRISA/TEXMEX

Juin 2008

# Introduction

- Base de données représentées par des vecteurs de  $\mathbb{R}^n$
- Similarité = Distance (euclidienne)
- But : trouver les  $k$  plus proches voisins d'un point requête.

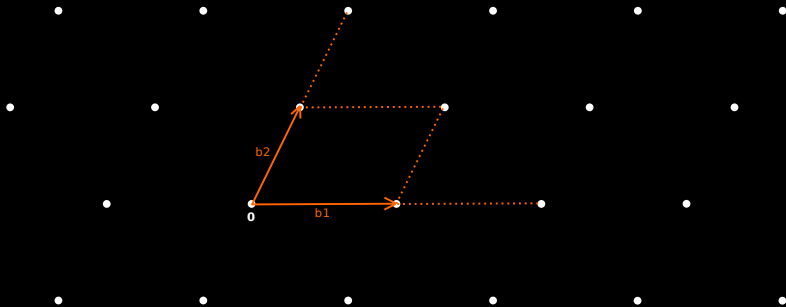
Algorithme naïf :

comparer la requête à toutes nos données,  
garder les  $k$  plus proches.

- **Indexation** des données : organisation, pre-traitement
- Recherche **Approximative**

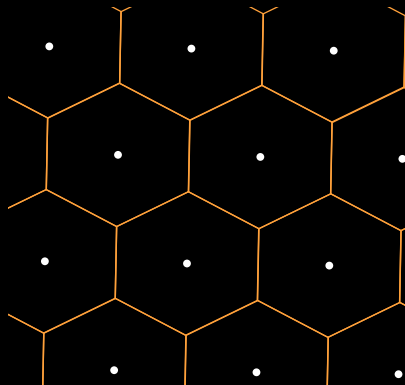
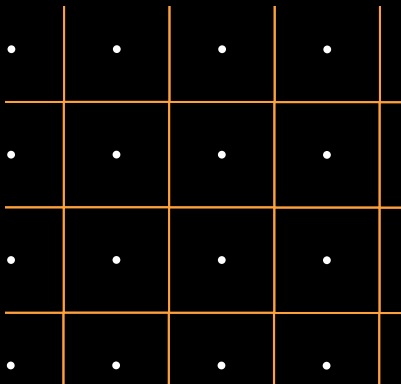
# Les réseaux géométriques

Un **réseau**  $\Lambda$  est un ensemble de points engendré par une base de vecteurs de  $\mathbb{R}^n$ .



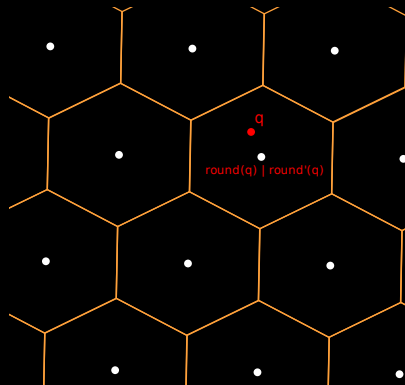
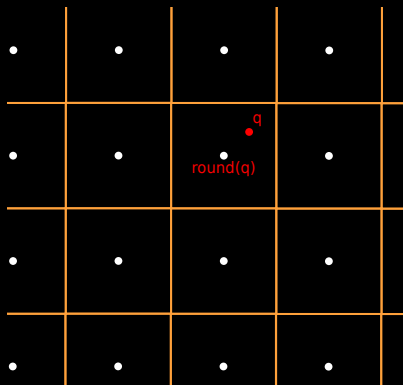
# Régions de Voronoï

- Les régions de Voronoï des points seront nos cellules
- Forme et volume identiques en tout point du réseau



# Décodage (Quantification vectorielle)

- Indépendant du nombre de données
- Indépendant du nombre de cellules

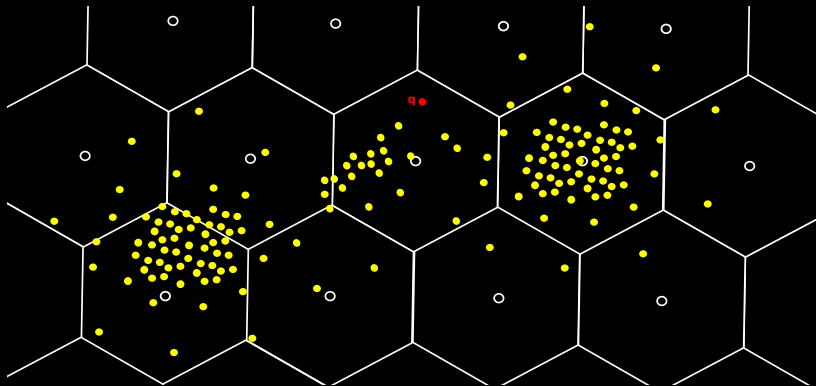


# Expériences

- 1 million de vecteurs Sift (images)
- Base en dimension 128
- Base en dimension 8
- Réseaux :  $\mathbb{Z}^n$ ,  $D_n$ ,  $D_n^+$ ,  $D_n^*$ ,  $A_n$ ,  $A_n^*$

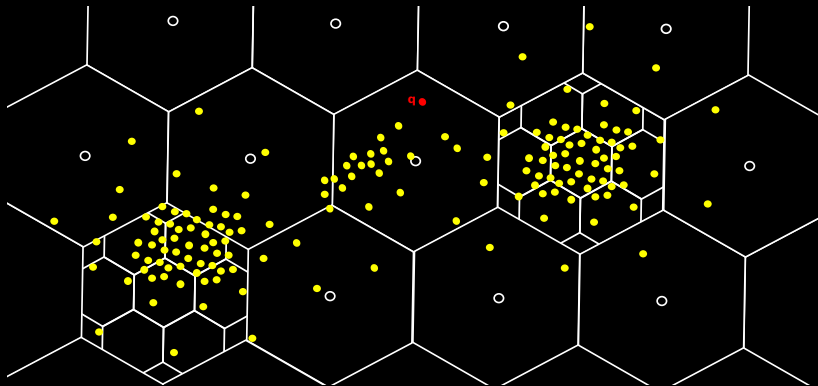
# Indexation avec un réseau

- Choix du réseau, échelle, translation, rotation
- Compromis peu de cellules vides / peu de cellules très pleines



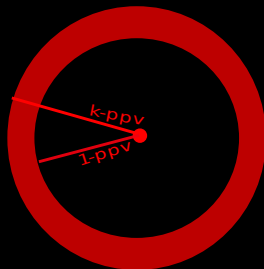
# Indexation avec un réseau

- Choix du réseau, échelle, translation, rotation
- Compromis peu de cellules vides / peu de cellules très pleines

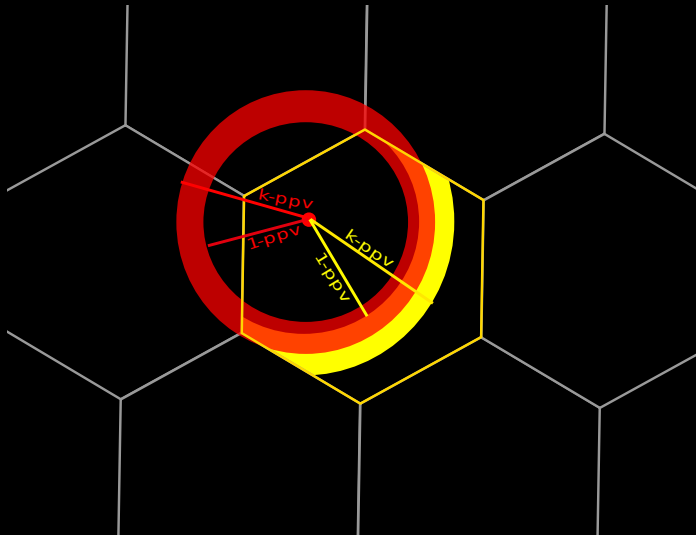




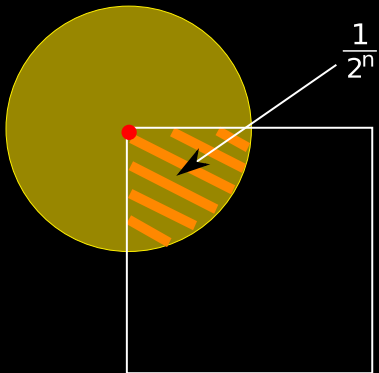
# Plus proches voisins



# Plus proches voisins dans un réseau



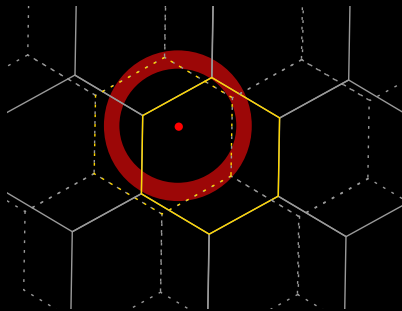
# Effet de la dimension



- $n = 2 \longrightarrow 0.25$
- $n = 128 \longrightarrow 3 \cdot 10^{-39}$

# Booster les résultats

Utiliser plusieurs réseaux :



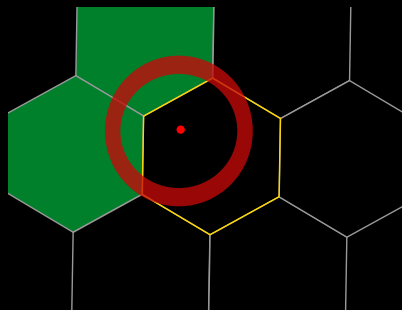
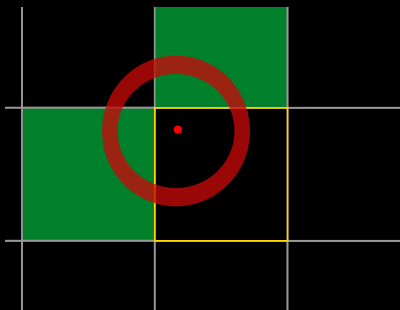
- Choisir le plus centré sur la requête, ou
- union des cellules.

Les données sont dupliquées dans chacun des réseaux.

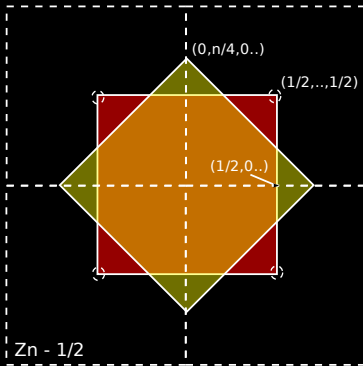
# Booster les résultats sans duplication

Profiter de la géométrie des réseaux :

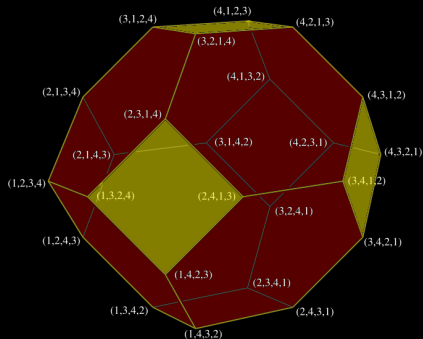
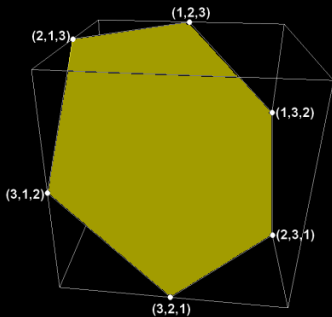
- Si on connaît une face, on sait calculer la cellule derrière
- Bon remplissage de la boule des plus proches voisins
- On veut un nombre raisonnable de faces, calculées rapidement



# L'Octaèdre tronqué ( $D_n^*$ )



# Le Permutoèdre ( $A_{n-1}^*$ )



$n!$  sommets,  $2^n - 2$  faces

Sommet le plus proche : même ordre dans les coordonnées

# Expériences

Rappel des 50 plus proches voisins sur 1000 requêtes :

- dimension 128,  $D^*$  :
  - réseau seul : 16% pour 1% lu
  - union : 66% pour 6% lu
  - ppf : 47% pour 7% lu
- dimension 8,  $A^*$ 
  - réseau seul : 40% pour 0.5% lu
  - union : 93% pour 3% lu
  - ppf : 87% pour 2.5% lu (66% pour  $D^*$ )



# Conclusions

Les résultats peuvent paraître décevant, mais

- on travaille directement dans la dimension des données,
- on utilise peu d'index
- cela s'améliore quand la dimension diminue

Approche connexe : LSH

- travail en dimension 1 (projections)
- des multitudes d'index

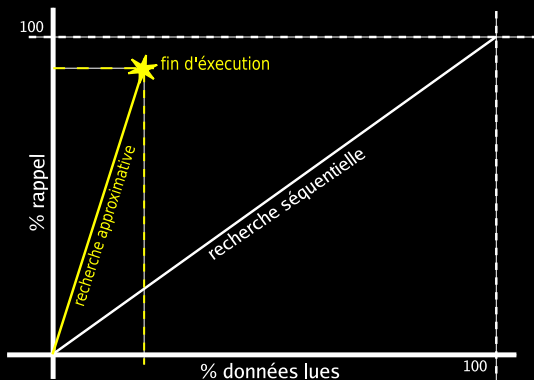
Réduire ce nombre d'index grâce aux réseaux :

- en travaillant en dimension  $> 1$
- en utilisant leur géométrie : plus proches faces.

# Questions ?

# Bonus

# Recherche approximative

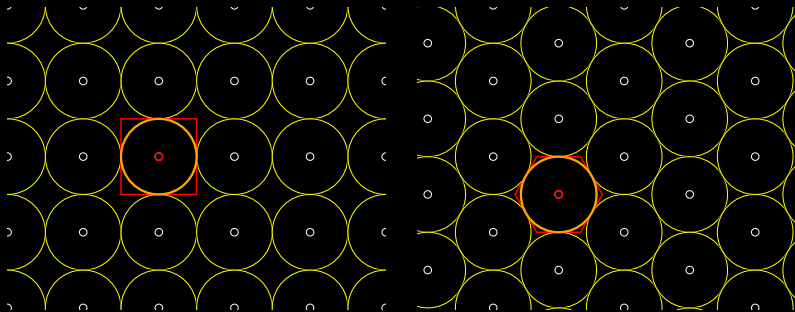


On veut cibler une petite région de l'espace contenant des potentiels plus proches voisins.

# Défis géométriques (1)

## Sphere packings

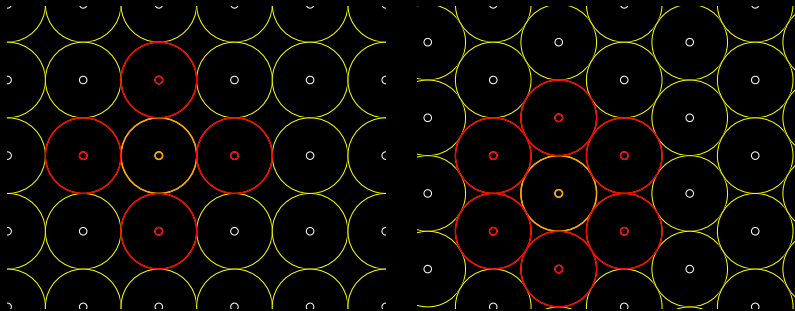
Empilement de sphères le plus compact possible



## Défis géométriques (2)

### Kissing number

Nombre d'embrassements maximal possible



# Défis géométriques (3)

## Coverings

Recouvrement de tout l'espace par des sphères

