Bases de données multimédia

LSH. Jointures par similarité.

Michel Crucianu (prenom.nom@cnam.fr)
http ://cedric.cnam.fr/~crucianm/bdm.html

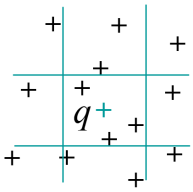
Département Informatique
Conservatoire National des Arts & Métiers, Paris, France

12 février 2018

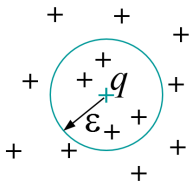
Plan du cours

Recherche par similarité

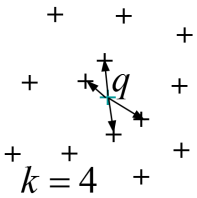
- Recherche par intervalle (*range query*) : $Range_r(\mathbf{q}) = \{\mathbf{x} \in \mathcal{D} | \forall i, |x_i - q_i| \leq r_i\}$
- Recherche dans un rayon (*sphere query*) : $Sphere_\epsilon(\mathbf{q}) = \{\mathbf{x} \in \mathcal{D} | d(\mathbf{x}, \mathbf{q}) \leq \epsilon\}$
- Recherche des k plus proches voisins ($k$ppv, *kNN*) :
  $kNN(\mathbf{q}) = \{\mathbf{x} \in \mathcal{D} \,|\, |kNN(\mathbf{q})| = k \wedge \forall \mathbf{y} \in \mathcal{D} - kNN(\mathbf{q}), d(\mathbf{y}, \mathbf{q}) > d(\mathbf{x}, \mathbf{q})\}$



range query



sphere query

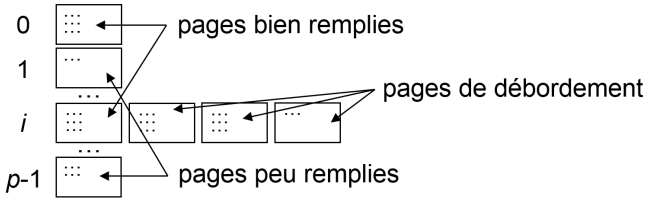


kNN query

N données $\Rightarrow$ complexité $O(N)$?

Hachage dans les bases de données relationnelles

- Objectif dans ce contexte : trouver directement où est stockée une donnée
- Principe (BD relationnelle) : p pages disque, attribut qui prend des valeurs v dans un domaine $\mathcal{D}$, fonction de hachage $h : \mathcal{D} \to \{0, 1, 2, \ldots, p-1\}$
- Exemple (BD relationnelle) :
  - Table `film(titre, realisateur, annee)`, hachage sur l'attribut `titre`
  - Fonction de hachage (de préférence, p nombre premier) : h(`titre`) = (somme_codes_ascii_caractères(`titre`)) modulo p

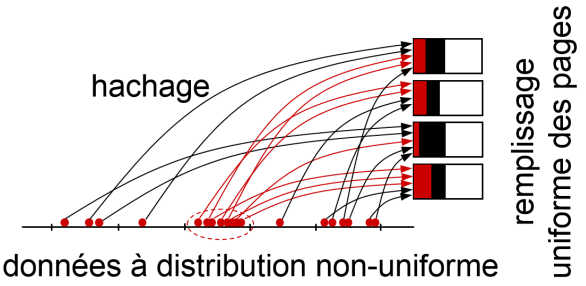


- Propriété recherchée : « disperser » uniformément des données dont la distribution peut être très non-uniforme au départ

Hachage dans les bases de données relationnelles (2)

- Recherche par identité : retrouver les autres informations concernant un film à partir de son titre
  - Hachage du titre à rechercher
  - Lecture de la page disque identifiée par le *hash*

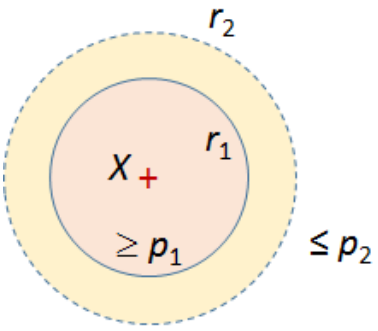
  $\Rightarrow$ complexité $O(1)$
- Dispersion uniforme de données non-uniformes $\rightarrow$ destruction de la structure de similarité des données $\Rightarrow$ méthode inadaptée à la recherche par intervalle ou par similarité



Hachage sensible à la similarité (*Locality-Sensitive Hashing, LSH*)

- Données dans un domaine $\mathcal{D}$, ensemble de *hash* (empreinte, condensé) $\mathcal{Q}$, métrique $d_{\mathcal{H}} : \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}^+$
- $\mathcal{H} = \{h : \mathcal{D} \rightarrow \mathcal{Q}\}$ est un ensemble de fonctions de hachage (r_1, r_2, p_1, p_2)-sensibles (avec $r_2 > r_1 > 0$, $p_1 > p_2 > 0$) si (voir par ex. [2])

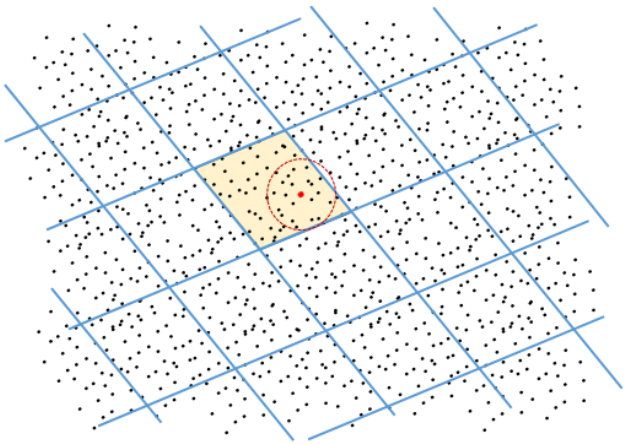
$$\forall x, y \in \mathcal{D}, \quad \begin{array}{lll} d_{\mathcal{H}}(x,y) \leq r_1 & \Rightarrow & P_{h \in \mathcal{H}} h(x) = h(y) \geq p_1 \\ d_{\mathcal{H}}(x,y) > r_2 & \Rightarrow & P_{h \in \mathcal{H}} h(x) = h(y) \leq p_2 \end{array} \tag{1}$$



Recherche par similarité avec LSH

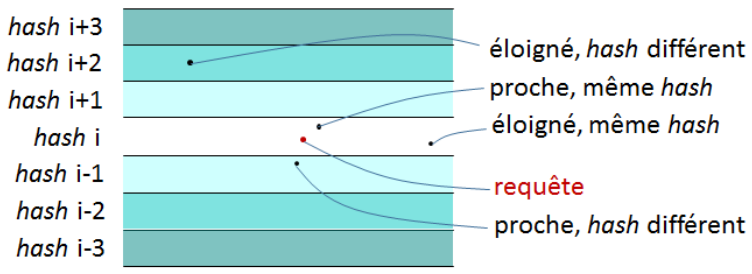
- Avant toute requête :
  1. Calculer la valeur de *hash* pour chacune des données de la base
  2. Stocker les données de même valeur de *hash* dans une même page (*bucket*)
- Pour chaque donnée-requête :
  1. Hachage de la donnée-requête, lecture de la page (*bucket*) associée au *hash*
  2. Retour des données de cette page
  3. Éventuellement, filtrage de ces données par calcul des distances

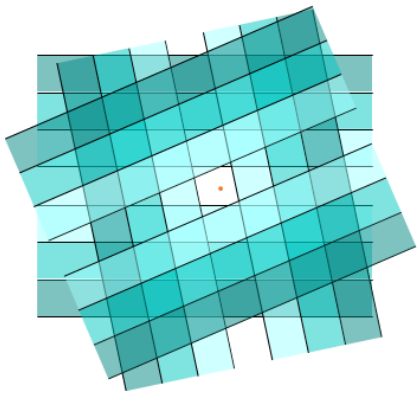
  $\Rightarrow$ Complexité $O(1)$



LSH pour la métrique euclidienne

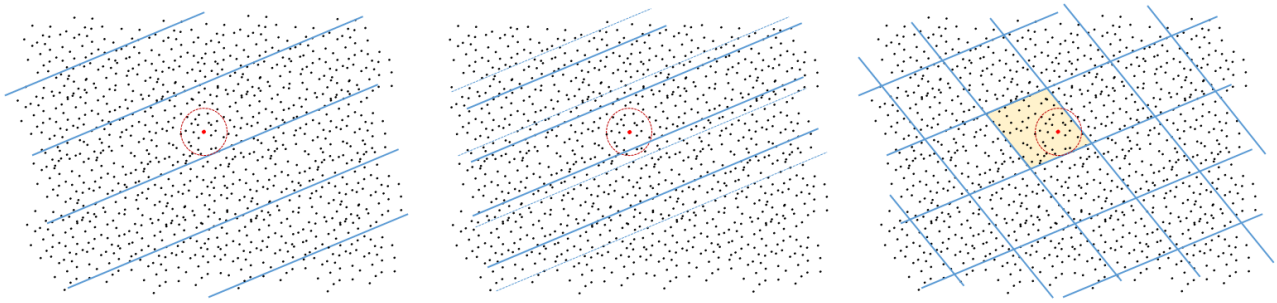
- $\mathcal{D} = \mathbb{R}^m$, $\mathcal{Q} = \mathbb{Z}$, $d_{\mathcal{H}}$ est la métrique L_2
- Fonctions élémentaires $h : \mathbb{R}^m \to \mathbb{Z}$, $h_{\mathbf{a},b,w}(\mathbf{x}) = \left\lfloor \frac{\mathbf{a}^T \cdot \mathbf{x} + b}{w} \right\rfloor$ avec $\mathbf{a} \in \mathbb{R}^m$ de composantes tirées indépendamment suivant $\mathcal{N}(0\,,1)$ et $b \in \mathbb{R}$ tiré suivant la loi uniforme dans $[0\,,1)$, $w \in \mathbb{R}$
- Table de hachage (ou fonction de hachage composée) : concaténation de (ET logique entre) n fonctions élémentaires indépendantes $\rightarrow$ *hash* de n entiers
- Exemple dans $\mathbb{R}^2$, avec 3 fonctions élémentaires :





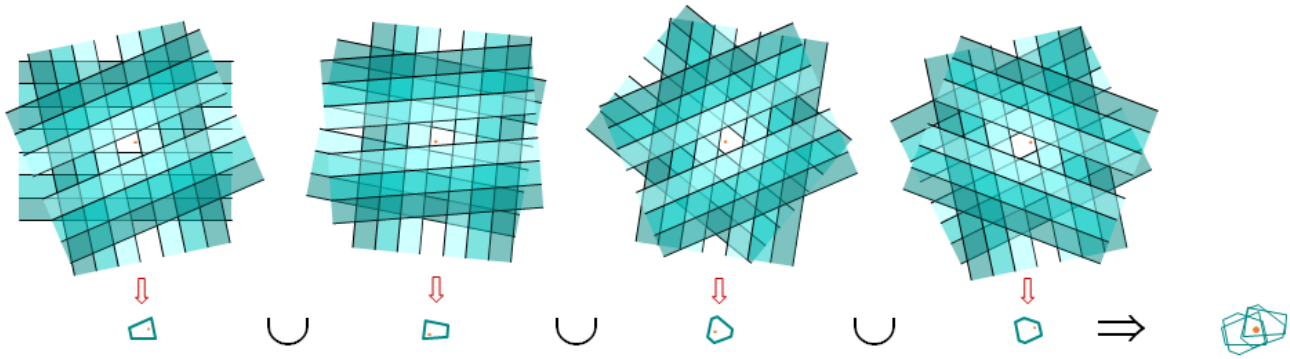
LSH : pourquoi regrouper des fonctions de hachage

- 1 fonction élémentaire n'est pas assez sélective ⇒ précision insuffisante (trop de faux positifs à filtrer ensuite par de coûteux calculs de distance)
- Grille plus fine ⇒ meilleure précision mais forte réduction du rappel
- Concaténation de (ET logique entre) n (> 1) fonctions de hachage indépendantes ($\rightarrow$ 1 table de hachage) ⇒ améliorer la précision sans diminuer trop fortement le rappel
- Attention, contrairement au cas présenté dans la figure, en général $n \ll$ dimension de l'espace !



LSH : pourquoi regrouper des tables de hachage

- 1 table de hachage ⇒ si requête proche d'une frontière alors des données similaires sont de l'autre côté ⇒ faux négatifs (réduction du rappel)
- Réunion (OU logique entre) des résultats issus de t (> 1) tables de hachage indépendantes ⇒ meilleur rappel



Distance cosinus

- Souvent, la description des données est hybride :
  1. Données multimédia : descripteurs d'images, vidéos, séquences audio
  2. Données textuelles : mots-clés, *tags*, textes $\Rightarrow$ représentation TF-IDF (par ex.)
  3. Données structurées : méta-données comme date, auteur, lieu, etc., représentées par des variables qualitatives et/ou ensembles
- Quelle distance employer ?
  - Distance cosinus : peut cumuler les contributions des différents types de descriptions (les ensembles seront représentés par leurs vecteurs caractéristiques)

    $\mathcal{D} = \mathbb{R}^m$, $d_{\cos}(\mathbf{x}, \mathbf{y}) = \arccos \frac{\mathbf{x}^T \cdot \mathbf{y}}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|}$
    - Cumul de plusieurs types de descriptions $\rightarrow$ choisir leurs pondérations relatives !

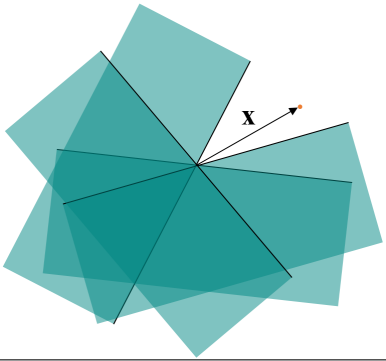
LSH pour la distance cosinus

- Fonctions élémentaires $h \in \mathcal{H}_{\cos}$, $h : \mathbb{R}^m \rightarrow \{0, 1\}$,

$$h_{\mathbf{v}}(\mathbf{x}) = \begin{cases} 1 & \mathbf{x}^T \cdot \mathbf{v} \geq 0 \\ 0 & \mathbf{x}^T \cdot \mathbf{v} < 0 \end{cases} \tag{2}$$

  avec $\mathbf{v} \in \mathbb{R}^m$ tiré suivant la loi uniforme sur l'hypersphère unité
- $\mathcal{H}_{\cos}$ est un ensemble de fonctions de hachage $(r_1, r_2, 1 - \frac{r_1}{180}, 1 - \frac{r_2}{180})$-sensibles

  Justification : $h_{\mathbf{v}}(\mathbf{x}) \neq h_{\mathbf{v}}(\mathbf{y})$ si l'hyperplan dont le vecteur normal est $\mathbf{v}$ passe entre $\mathbf{x}$ et $\mathbf{y}$; cela arrive avec une probabilité $\frac{d_{\cos}(\mathbf{x},\mathbf{y})}{180}$ $\Rightarrow$ probabilité de collision est $1 - \frac{d_{\cos}(\mathbf{x},\mathbf{y})}{180}$
- Exemple dans $\mathbb{R}^2$, avec 4 fonctions élémentaires :



Similarité entre ensembles finis

- Indice (ou similarité) de Jaccard :
  - Soit un ensemble $\mathcal{E}$, $\mathcal{D} = \mathcal{P}(\mathcal{E})$ est l'ensemble des parties de $\mathcal{E}$
  - Similarité entre deux sous-ensembles $\mathcal{A}, \mathcal{B} \in \mathcal{P}(\mathcal{E})$: $s_J(\mathcal{A}, \mathcal{B}) = \frac{|\mathcal{A} \cap \mathcal{B}|}{|\mathcal{A} \cup \mathcal{B}|}$
  - $d_J(\mathcal{A}, \mathcal{B}) = 1 - s_J(\mathcal{A}, \mathcal{B})$ est une métrique sur $\mathcal{P}(\mathcal{E})$
- Définir les ensembles :
  - Texte : mots
  - Profil d'achat : ensemble d'articles achetés
- Pour réduire l'ordre de complexité de la recherche par similarité (éviter de comparer une requête à chacune des N données de la base) : définir des fonctions LSH adaptées et s'en servir pour une recherche $O(1)$

LSH pour ensembles

- Fonctions de hachage élémentaires : $h_\pi : \mathcal{P}(\mathcal{E}) \to \mathcal{E}$, $h_\pi(\mathcal{A}) = \min \pi(\mathcal{A})$
  - On fixe un ordre des éléments de $\mathcal{E}$
  - π est une permutation des éléments de $\mathcal{E}$
  - $\min \pi(\mathcal{A})$ est l'élément de $\mathcal{A}$ qui se retrouve premier après cette permutation
- Représentation des ensembles par leurs fonctions caractéristiques :
  - Exemple de permutation : $\pi = \begin{pmatrix} a & b & c & d & \dots \\ c & d & a & b & \dots \end{pmatrix}$

$\mathcal{E}$	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	$\pi(\mathcal{E})$	$\pi(\mathcal{A})$	$\pi(\mathcal{B})$	$\pi(\mathcal{C})$
a	1	0	0	c	1	1	0
b	0	0	1	d	0	0	0
c	1	1	0	a	1	0	0
d	0	0	0	b	0	0	1
...				...			

LSH pour ensembles (2)

- Probabilité de collision : pour toute fonction h_π et quels que soient les ensembles $\mathcal{A}, \mathcal{B} \in \mathcal{P}(\mathcal{E})$

$$p(h_\pi(\mathcal{A}) = h_\pi(\mathcal{B})) = s_J(\mathcal{A}, \mathcal{B})$$

- Justification :
  - Soit x le nombre d'éléments communs entre $\mathcal{A}$ et $\mathcal{B}$ (c'est à dire $|\mathcal{A} \cap \mathcal{B}|$)
  - Soit y le nombre d'éléments spécifiques à $\mathcal{A}$ ou $\mathcal{B}$ (c'est à dire $|(\mathcal{A} - \mathcal{B}) \cup (\mathcal{B} - \mathcal{A})|$)
  - Alors $s_J(\mathcal{A}, \mathcal{B}) = \frac{x}{x+y}$
  - Pour toute fonction h_π, la probabilité de trouver en premier après la permutation π un élément commun plutôt qu'un élément spécifique est $\frac{x}{x+y}$
  - Donc $p(h_\pi(\mathcal{A}) = h_\pi(\mathcal{B})) = \frac{x}{x+y}$

LSH pour ensembles (3)

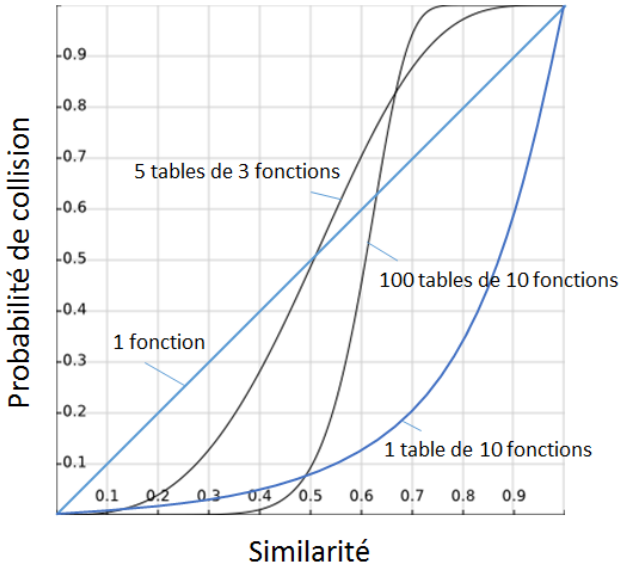
1. Construction d'une table de hachage en regroupant n fonctions de hachage choisies aléatoirement de façon indépendante :
   - Le *hash* donné par la table est la concaténation des *hash* des n fonctions
   - 2 données sont en collision dans la table $\Leftrightarrow$ elles sont en collision par rapport à la première fonction de hachage ET par rapport à la deuxième ET ... ET par rapport à la n-ème

   $\Rightarrow$ probabilité de collision dans la table : s^n (s étant la similarité $s_J(\mathcal{A}, \mathcal{B})$)
2. Utilisation de *plusieurs* (t) tables de hachage :
   - 2 données sont en collision si elles sont en collision dans la première table OU dans la deuxième OU ...OU dans la t-ème

   $\Rightarrow$ probabilité de collision dans au moins une table : $1 - (1 - s^n)^t$
   Justification : probabilité de non collision dans 1 table $= 1 - s^n$ $\rightarrow$ probabilité de non collision dans les t tables $= (1 - s^n)^t$ $\rightarrow$ probabilité de collision dans au moins 1 table $= 1 - (1 - s^n)^t$

LSH pour ensembles (4)

- Graphique probabilité de collision ($\in (0,1)$) fonction de $s_J(\mathcal{A},\mathcal{B})$ ($\in (0,1)$) :



Généralisation : « amplification » de fonctions *LSH*

- Fonctions de hachage (r_1, r_2, p_1, p_2)-sensibles (avec $r_2 > r_1 > 0$, $p_1 > p_2 > 0$) :

$$\forall x, y \in \mathcal{D}, \quad \begin{array}{lll} d_{\mathcal{H}}(x,y) \leq r_1 & \Rightarrow & P_{h\in\mathcal{H}} h(x) = h(y) \geq p_1 \\ d_{\mathcal{H}}(x,y) > r_2 & \Rightarrow & P_{h\in\mathcal{H}} h(x) = h(y) \leq p_2 \end{array} \tag{3}$$

- Amplification : rapprocher p_2 (probabilité de collision pour données dissimilaires) de 0 et p_1 (probabilité de collision pour données similaires) de 1
  - ET logique entre n fonctions : $h_{\text{AND}}(x) = h_{\text{AND}}(y)$ si et seulement si $h_i(x) = h_i(y)$ pour tout i, $1 \leq i \leq n \Rightarrow h_{\text{AND}}$ est (r_1, r_2, p_1^n, p_2^n)-sensible
    - → justifie le regroupement de plusieurs fonctions dans une table
  - OU logique entre t fonctions : $h_{\text{OR}}(x) = h_{\text{OR}}(y)$ si et seulement si $h_i(x) = h_i(y)$ pour au moins une valeur de i, $1 \leq i \leq n \Rightarrow h_{\text{OR}}$ est $(r_1, r_2, 1-(1-p_1)^t, 1-(1-p_2)^t)$-sensible
  - $h_{\text{OR,AND}}$ est $(r_1, r_2, 1-(1-p_1^n)^t, 1-(1-p_2^n)^t)$-sensible
    - → justifie l'utilisation conjointe de plusieurs tables

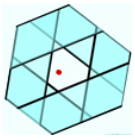
Extensions et évolutions de *LSH*

1. Fonctions de hachage indépendantes de la distribution des données
   - Extensions à d'autres métriques ou mesures de similarité : $\chi 2$, noyaux, configurations...
   - Améliorations de l'efficacité de la recherche : *multi-probe*, *a posteriori multi-probe*...
2. Fonctions de hachage qui exploitent la distribution des données pour améliorer la sélectivité
   1. Sans supervision : *Spectral Hashing*, *Random Maximum Margin Hashing*...
   2. Avec supervision : augmenter la probabilité de collision des données appartenant à une même classe et la diminuer pour les données de classes différentes

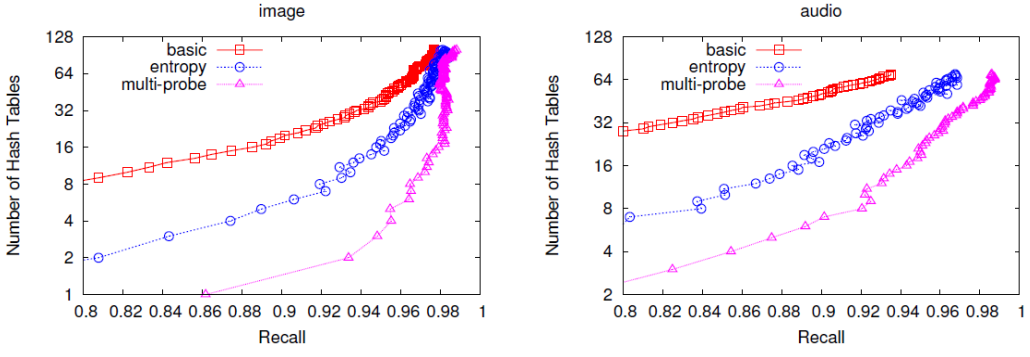
(voir par ex. [6] pour une synthèse)

Multi-probe LSH

- Un bon rappel exige de nombreuses tables ⇒ occupation mémoire élevée !
- *Multi-probe LSH* [4] : pour réduire le nombre de tables nécessaires, dans chaque table échantillonner aussi les *buckets* voisins



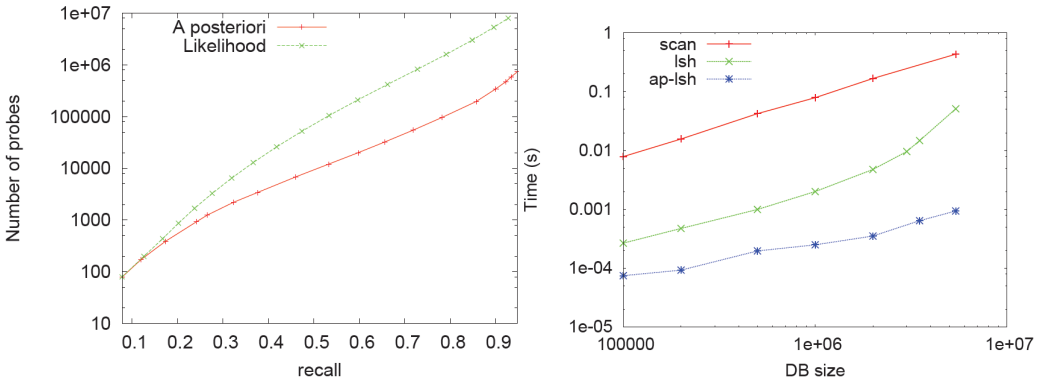
- Comparaisons :



⇒ réduction de l'espace mémoire nécessaire

A posteriori Multi-probe LSH

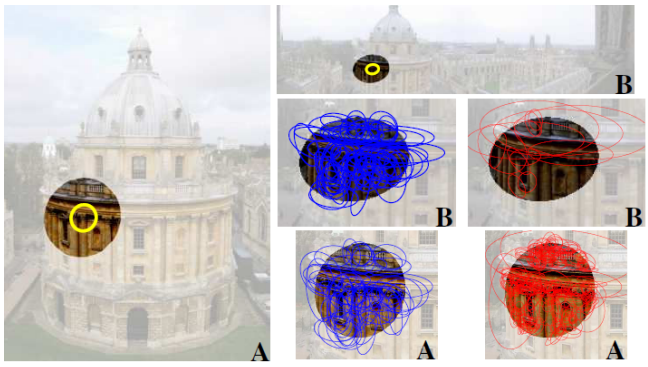
- Multi-probe LSH [4] : la définition des fonctions de hachage définit le voisinage d'un bucket B (ensemble des buckets à visiter quand le *hash* de la requête correspond à B)
- *A posteriori multi-probe LSH* [3]
  - Estimer la probabilité *a posteriori* de trouver des réponses pertinentes dans chaque bucket voisin
  - Choisir le nombre minimal de buckets tels que la somme de leurs probabilités dépasse un seuil de rappel α



⇒ réduction du temps de traitement d'une requête

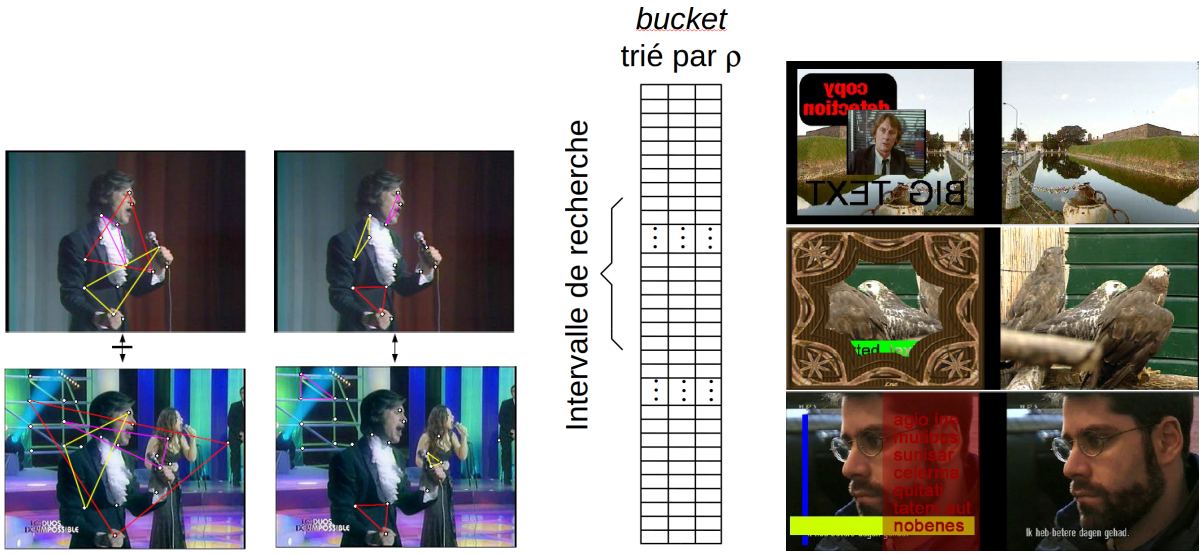
Min-hachage « géométrique »

- [1] : application de LSH pour ensembles à la comparaison d'ensembles de points d'intérêt
  - Information de co-occurrence de points → meilleure précision
  - Ensembles de points voisins → meilleure robustesse aux occultations partielles → meilleur rappel
- Construction d'un ensemble : sélection des points en utilisant l'information d'échelle issue du descripteur (SIFT) du point central



Min-hachage géométrique

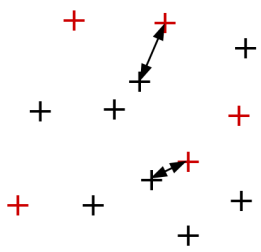
- [5] : inclusion d'une information géométrique (en plus de la proximité spatiale) ⇒ meilleur compromis entre rappel et précision
  - Quantification de l'espace de description en cellules → chaque bucket est défini par un triplet de cellules
  - Triplets de points définis par les kppv dans l'image
  - Information géométrique : $\rho = \frac{d(q,ppv1)}{d(q,ppv2)}$, tri de chaque bucket par ρ croissant



Plan du cours

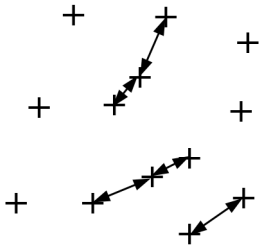
Jointure par similarité

- Jointure avec seuil de distance θ, ensembles $\mathcal{D}_1$, $\mathcal{D}_2$, avec N_1 et respectivement N_2 éléments : $K_\theta = \{(\mathbf{x}, \mathbf{y}) | \mathbf{x} \in \mathcal{D}_1, \mathbf{y} \in \mathcal{D}_2, d(\mathbf{x}, \mathbf{y}) \leq \theta\}$
- Auto-jointure : $\mathcal{D}_1 \equiv \mathcal{D}_2$



jointure par similarité

$\Rightarrow$ complexité $O(N_1 \times N_2)$?



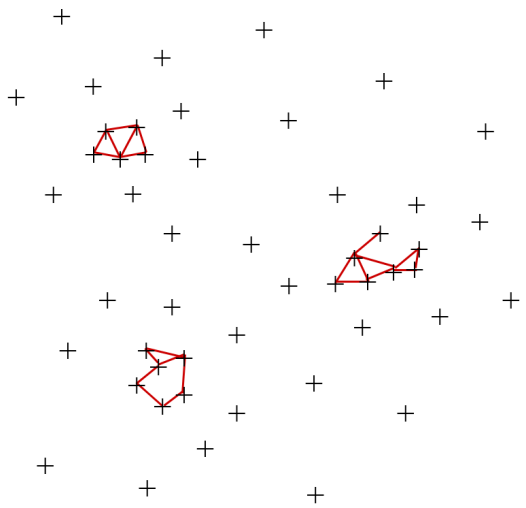
auto-jointure par similarité

$\Rightarrow$ complexité $O(N^2)$?

(Auto-)jointure par similarité

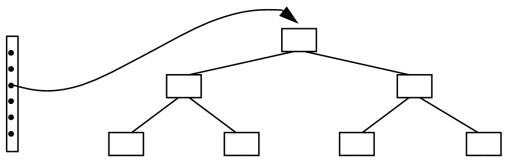
- Auto-jointure avec seuil de distance θ, ensemble de N données $\mathcal{D}$:

$$K_\theta = \{(\mathbf{x}, \mathbf{y}) | \mathbf{x}, \mathbf{y} \in \mathcal{D}, d(\mathbf{x}, \mathbf{y}) \leq \theta\} \quad (4)$$



Jointure par similarité : approches

1. Application directe de la recherche par similarité
   - Chaque objet devient une requête
   - Arbre $\rightarrow O(N \log N)$, LSH $\rightarrow O(N)$

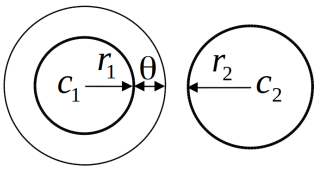


2. Regrouper le travail pour objets-requête similaires
   - Parcours en parallèle de 2 arbres $\rightarrow O(N \log N)$ (mais construction d'arbre $O(N \log N)$)
   - Partitionnement et jointure intra-partitions, méthodes exactes ou approximatives (par ex. LSH)

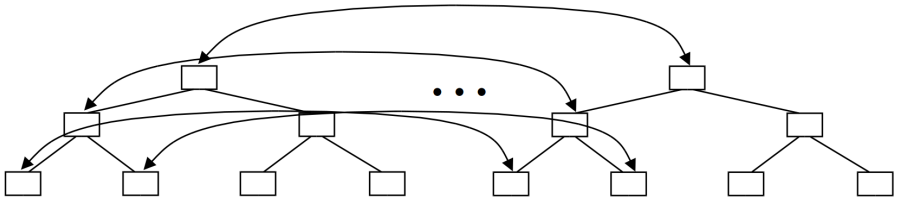
(rappel : ordre de complexité $\geq$ taille du résultat !)

Approche par parcours d'arbre(s)

- Idée : parcourir en parallèle les 2 arbres de recherche (un pour chaque ensemble)
  - Appliquer le test d'exclusion de 2 sous-arbres $d(c_1, c_2) > r_1 + r_2 + \theta$



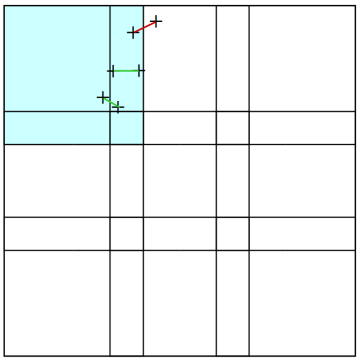
- Jointure $\rightarrow$ algorithme doublement récursif avec deux arbres :



- Auto-jointure $\rightarrow$ algorithme doublement récursif avec un seul arbre

Approche par partitionnement

- Idée (pour l'auto-jointure) : éviter de comparer des données trop éloignées
- Méthode : partitionnement du domaine en k parties, ensuite (auto-)jointure intra-partitions $\Rightarrow$ complexité $O(N^2) \searrow O(N^2/k)$



- Méthode exacte : superposition nécessaire entre partitions voisines, dépendant de θ $\rightarrow$ redondance, qui augmente rapidement avec la dimension
- Méthodes approximatives : pas de redondance $\Rightarrow$ rappel < 1
  - Si partitionnement fin, quels que soient 2 points d'une même partition, leur distance est $< \theta \Rightarrow$ l'étape de jointure intra-partitions est inutile

Auto-jointure par similarité avec *LSH*

1. Choix de fonctions de hachage suivant la distance utilisée, réglage de l'« amplification » en fonction du seuil de distance θ, ainsi que du coût de calcul et de stockage qui augmentent avec n (nombre de fonctions par table de hachage) et avec t (nombre de tables) $\rightarrow h_{\texttt{OR},\texttt{AND}}$
2. Application de la fonction de hachage à toutes les données ; pour chaque valeur de $h_{\texttt{OR},\texttt{AND}}$, les données pour lesquelles le *hash* a cette valeur forment un *bucket*
3. Pour chaque *bucket* non vide, calcul des distances $d(\mathbf{x}, \mathbf{y})$ uniquement entre données du *bucket*

$\Rightarrow$ Évite de calculer la distance entre des données qui ne sont pas suffisamment similaires

$\Rightarrow$ Complexité : $O(N)$ pour l'étape (2) ; pour l'étape (3), borne inférieure qui dépend de la taille du résultat

Application à la détection de copies vidéo

- Détection de copies vidéo par le contenu (DCPC)
  - Qu'est-ce qu'une copie vidéo ?
  - Détection de copies par le contenu et recherche par similarité
  - Quelle description vidéo pour la détection de copies ?
  - Problème du passage à l'échelle et solutions
- Application : fouille de bases vidéo par auto-jointure par similarité pour DCPC
  - Fouille par DCPC et auto-jointure par similarité
  - Description compacte des images-clés
  - Indexation pour l'auto-jointure par similarité
  - Évaluation de la méthode et illustration des résultats

Détection de copies vidéo

- Copie = version transformée d'un contenu original
- Transformations rencontrées
  - Photométriques : contraste, gamma, passage en NB, bruit, flou
  - Géométriques : recadrage, changement d'échelle, étirement
  - Temporelles : changement de tempo, ajout et suppression d'images
  - Post-production : extraits, compilations, ajout de logos, de sous-titres, incrustations, bandes noires, ré-encodage, etc.
- Images-clés illustrant des exemples de copies transformées :


Détection de copies vidéo (2)

- Intérêt de la détection de copies
  - Protection des droits : détection de transmissions potentiellement illicites dans des flux vidéo (hertziens, satellitaires, par câble, Internet) ou dans des bases vidéo (Web2.0, pair-à-pair)
  - Suivi des contrats de diffusion et/ou facturation automatique : toute retransmission d'un extrait est facturée automatiquement, de façon précise (durée)
  - Fouille vidéo basée sur la détection de copies
- Approches :
  1. Détection de copies par tatouage (watermarking)
     - Uniquement pour originaux tatoués avant toute transmission...
     - Robustesse variable aux différents types d' « attaques »
     - Nombreuses méthodes de tatouage ⇒ surveillance difficile
  2. Détection par le contenu : pour garder une partie de l'intérêt du contenu original, la copie doit conserver l'essentiel de « l'information visuelle » présente

Détection de copies et recherche par similarité

- « Conserver l'essentiel de l'information visuelle » ⇒ similarité (pas n'importe laquelle !) entre l'original et la « copie »
- Principe de détection de copie vidéo par le contenu
  - *La vidéo candidate est une copie de l'original si elle est suffisamment similaire (avec une représentation adéquate du contenu et une mesure de similarité adaptée) à l'original*

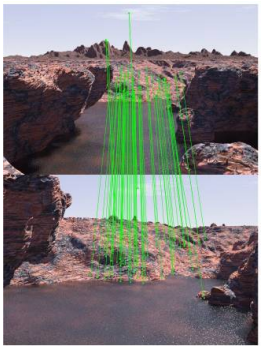
  → La représentation de la vidéo candidate est utilisée comme une requête par similarité dans une base contenant les représentations des vidéos originales

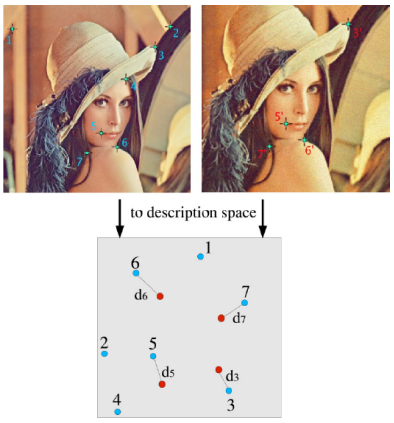
Quelle description vidéo pour la détection de copies par le contenu ?

- Description globale des images ?
  - Peu robustes aux transformations attendues :


→ Description locale des images
- SIFT, PCA-SIFT, GLOH, SURF ?
  - Trop invariants aux changements d'échelle, angle de vue, etc. ?
  - Très coûteux (extraction, recherche)

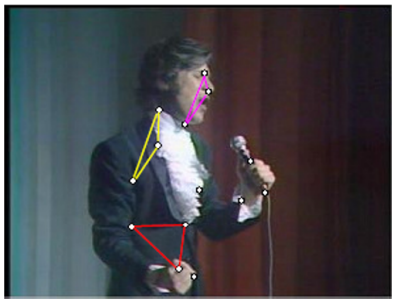
→ Détecteur amélioré de Harris, description différentielle
- Robustesse (dans des limites assez étroites) aux changements d'échelle, contraste
- Assez léger (vecteurs de dimension 20)





Fouille par détection de copies et auto-jointure par similarité

- Objectif : trouver des séquences vidéo dupliquées avec transformations
- Données : trames des vidéos d'une grande base
- Description de chaque trame : extraction et description de « points d'intérêt »
- Similarité : configuration géométrique de triplets de points d'intérêt
- Fonctions LSH : min-hachage géométrique [5]



Résultats d'auto-jointure par similarité avec LSH

- Précision et rappel sur vérités terrain (durée < 1 minute !) :
  - Base CIVR 2007 (env. 80 h) : précision 0,96 pour rappel 0,8
  - Base INA (env. 30 h) : précision 0,95 pour rappel 0,84
- Rapidité sur plus grandes bases :

Taille base	2000 h	5000 h	10000 h
Nombre de trames-clé	$5,8 \times 10^6$	$14,5 \times 10^6$	$28,7 \times 10^6$
Construction base	2h 35min	3h 38min	7h 00min
Création liens entre trames	5h 40min	14h 59min	55h 00min
Création liens entre séquences	1h 15min	7h 15min	20h 35min

Illustration de résultats d'auto-jointure par similarité avec LSH

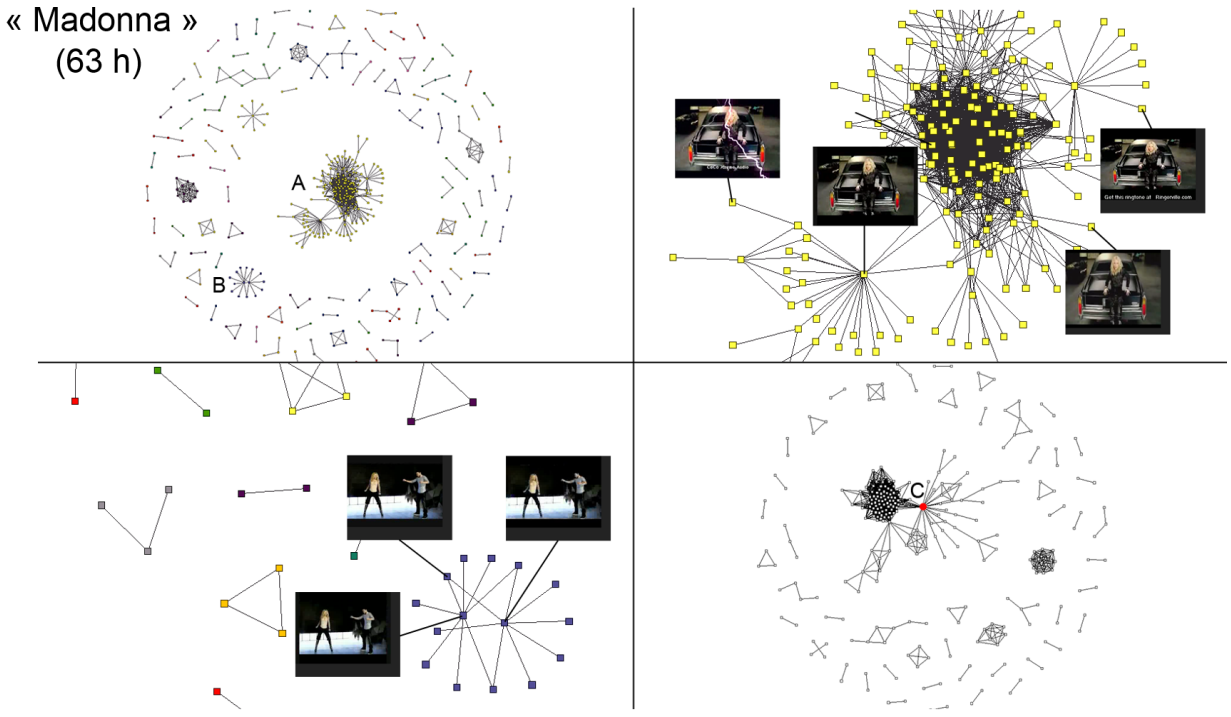
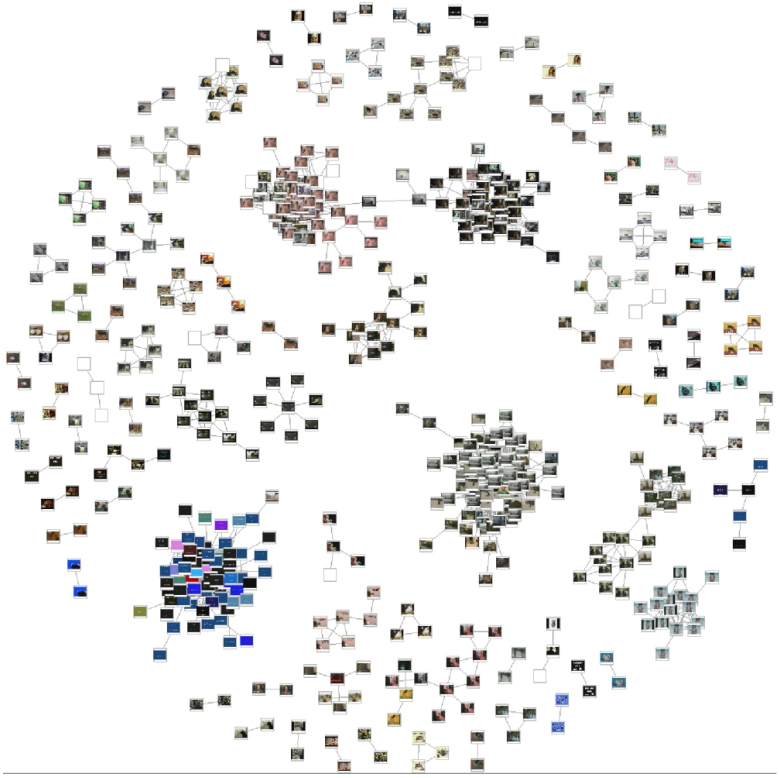


Illustration de résultats d'auto-jointure par similarité avec LSH (2)

« *funny cats* » (36 h)



Références I

O. Chum, M. Perd'och, and J. Matas.
Geometric min-hashing : Finding a (thick) needle in a haystack.
In *CVPR'09 : IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 17–24, June 20–26 2009.

A. Gionis, P. Indyk, and R. Motwani.
Similarity search in high dimensions via hashing.
In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 518–529, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

A. Joly and O. Buisson.
A posteriori multi-probe locality sensitive hashing.
In *MM '08 : Proceeding of the 16th ACM international conference on Multimedia*, pages 209–218, New York, NY, USA, 2008. ACM.

Références II


Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li.
Multi-probe LSH : Efficient indexing for high-dimensional similarity search.
In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 950–961. VLDB Endowment, 2007.


S. Poullot, M. Crucianu, and S. Satoh.
Indexing local configurations of features for scalable content-based video copy detection.
In *LS-MMRM : 1st Workshop on Large-Scale Multimedia Retrieval and Mining, in conjunction with 17th ACM international conference on Multimedia*, pages 43–50, New York, NY, USA, 2009. ACM.


J. Wang, H. T. Shen, J. Song, and J. Ji.
Hashing for similarity search : A survey.
CoRR, abs/1408.2927, 2014.