

Analyse et indexation d'images et de vidéos dans des grandes bases Multimédia

Cours n°3

2013-2014

Frederic Precioso

*Ce cours s'appuie sur les cours de George Bebis et de Derek Hoiem
et d'autres collègues*

Bag of Word : introduction

- Back to our target retrieval problem
 - Requires to find the same objects with deformations
 - Winning combination: sparse detector PoI + discriminant and invariant descriptors, such as SIFT
 - Local => so intrinsically robust to occlusions and noise (of the background)
 - Challenge: fast search of nearest neighbors in very large datasets
 - Geometrical consistency to constrain correspondances

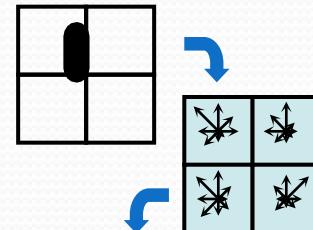
Database of Images



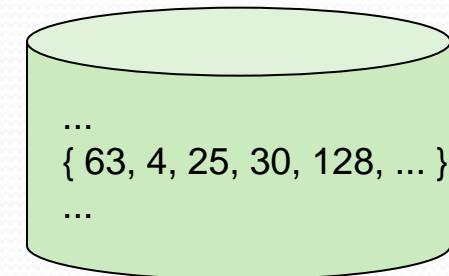
Find the Pol of Each Image



Describe Each Pol



Index all Descriptors



Offline

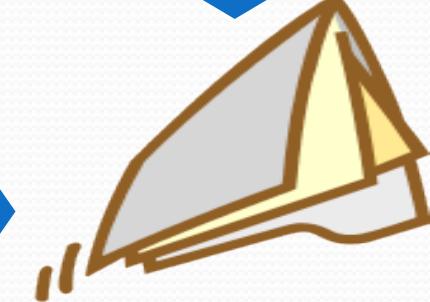
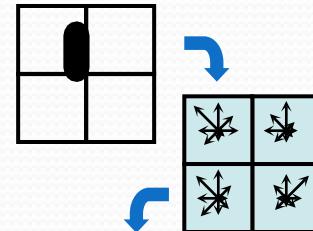
Query Image



Find the Pol

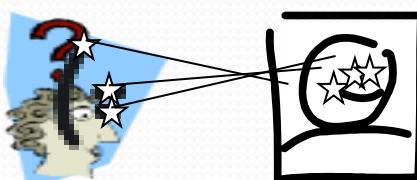


Describe Each Pol



Descriptor Matching

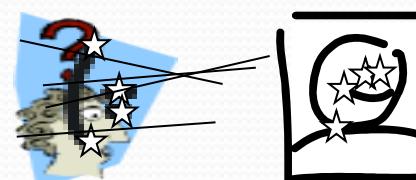
Refined Results



Geometric Consistency



Brute Results



Bag of Word: introduction

- Target retrieval efficient with local descriptor matching
 - But not scalable for (very) large datasets or very large images
 - Approximation required
- Categorization: we do not look for “identical” images
 - Model providing state-of-art results: “**Bag of Word**”
 - Text-mining approach

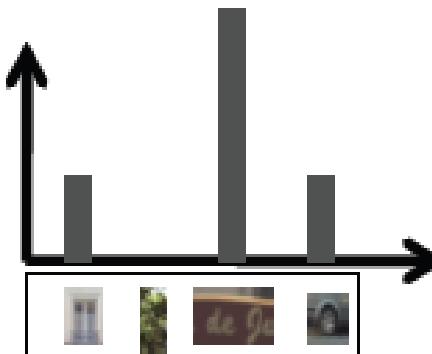
From Bag of Features (BoF) to (BoW)

- BoF: Local descriptor ensembles
 - How to compare two images ? => signature vectorielle
 - Trade-off discrimination / generalization

Sac de descripteurs
(features)



BoW : histogramme
sur un dictionnaire



Bag of Visual Words (BoW)

Image

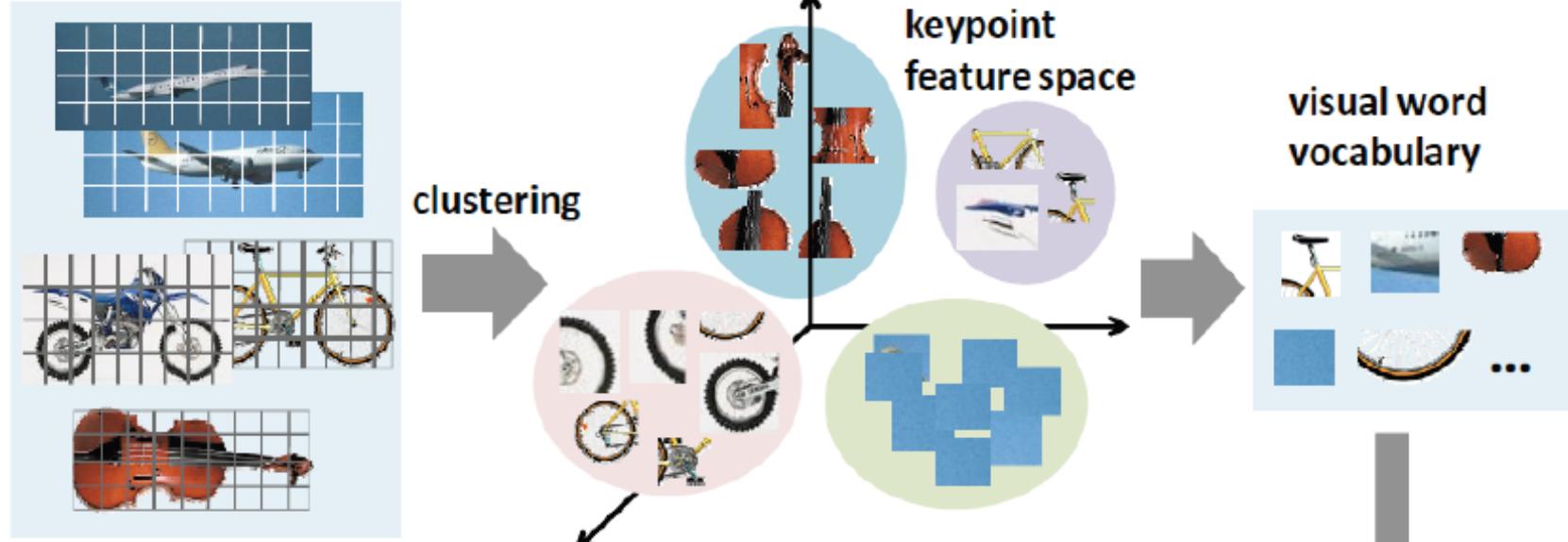


Bag of Visual
'Words'

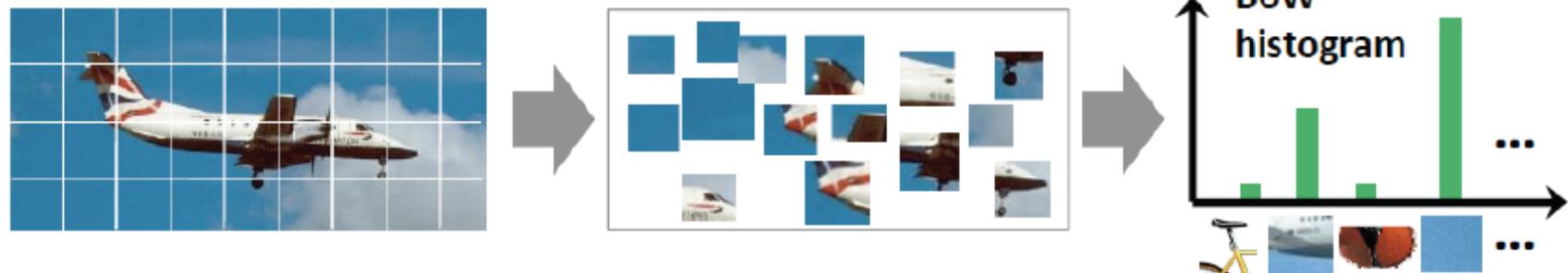


Bag of Visual Words (BoW)

offline

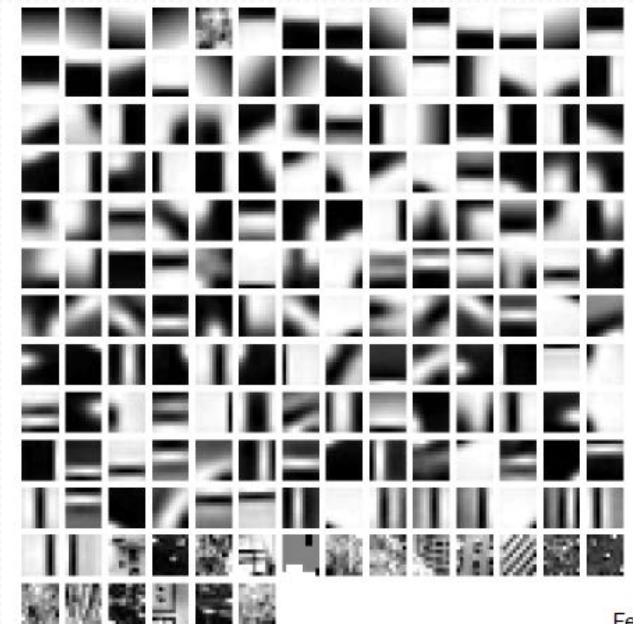
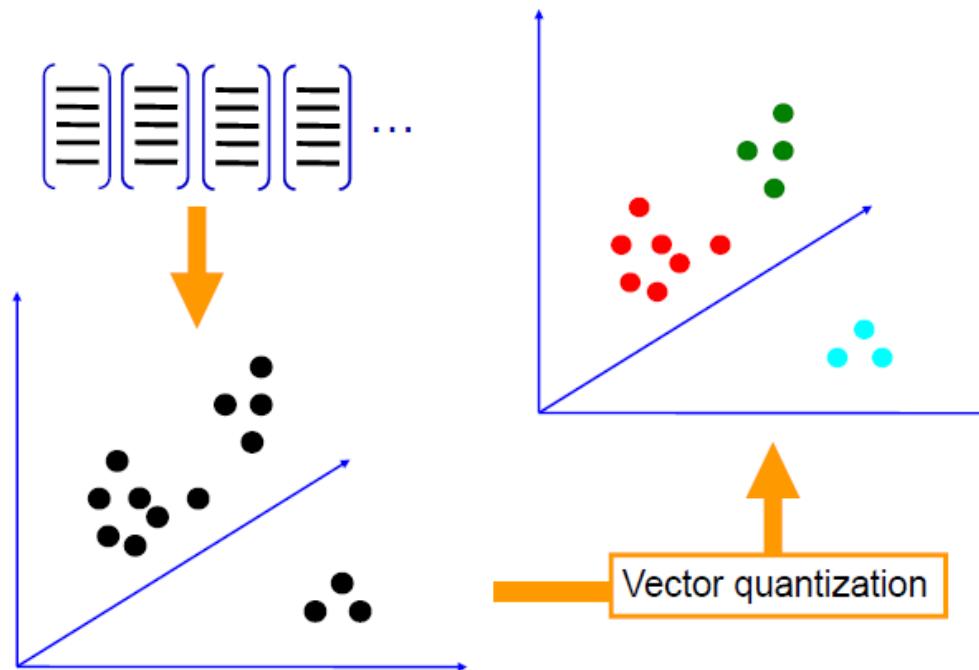


online



Bag of Visual Words: off-line part

- Descriptor extraction, visual dictionary based approaches
 - Construction of a dictionary of visual patterns (words) adapted to the dataset, e.g. K-Means



Le clustering

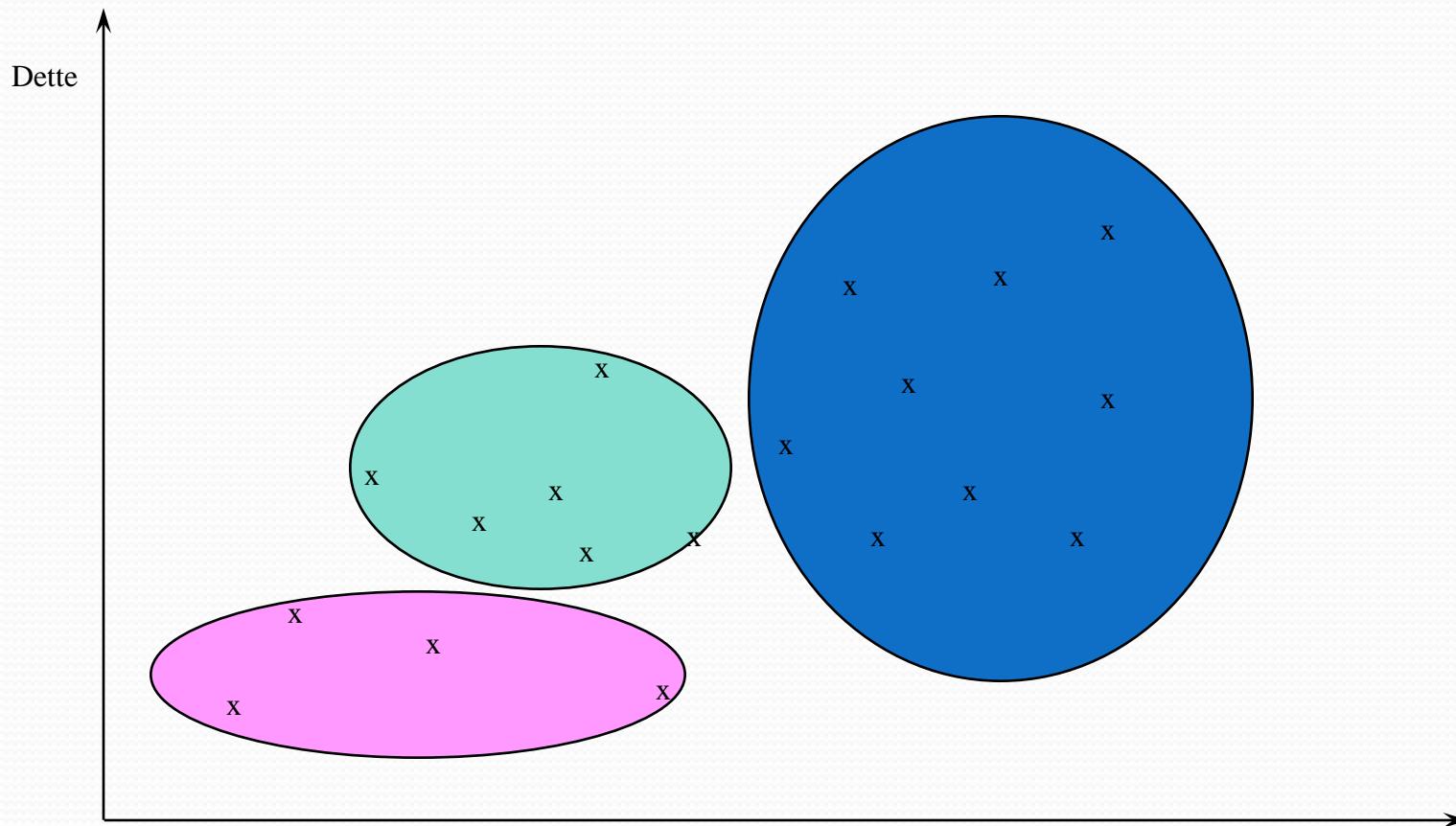
- Le clustering groupe des objets en se basant sur leurs similarités.
- La mesure de similarité peut être calculée pour différents types de données.
- La sélection de la mesure de similarité dépend des données utilisées et le type de similarité recherchée.

Clustering (partitionnement) = Apprentissage non supervisé

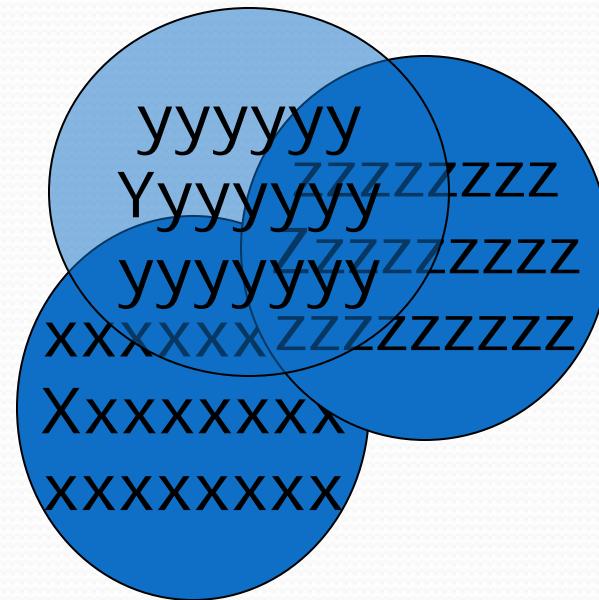
- Apprentissage non supervisé : les données ne sont pas classées, on isole des sous-groupes d'enregistrements similaires les uns aux autres
- Un fois les clusters détectés, on pourra appliquer des techniques de modélisation sur chaque cluster

Clustering

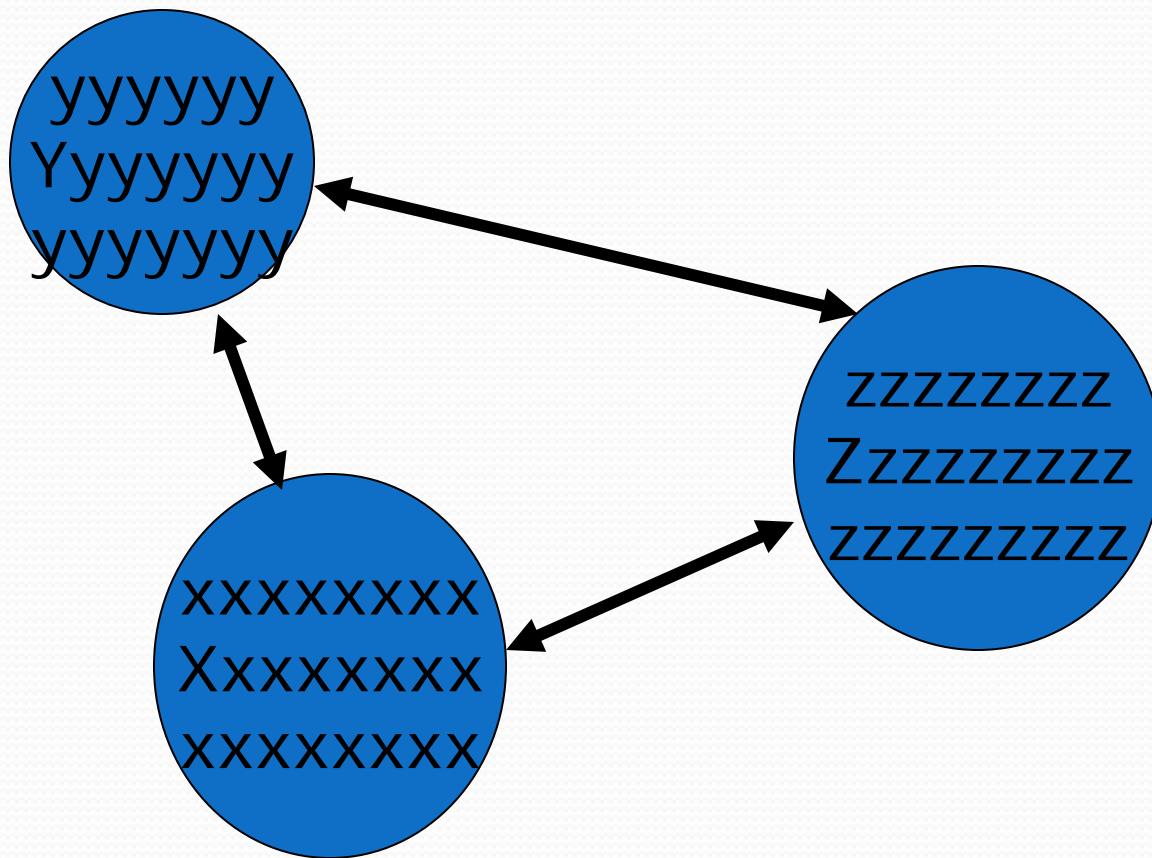
Pas d'affectation à une classe connue au départ : on regroupe les individus par leur proximité en classes qui peuvent se recouper.



Comment déterminer un bon clustering ?



Comment déterminer un bon clustering ?



Comment déterminer un bon clustering ?

- Un bon algorithme de classification fera en sorte qu'il y aura une :
 - petite variabilité intra-classe (c-à-d petite distance entre les individus d'un même groupe)
 - grande variabilité inter-classe (c-à-d grande distance entre les individus de groupes différents)
- La qualité des résultats de la classification dépendra de la mesure de distance utilisée et de l'algorithme choisi pour l'implémenter.

Notions de distance et de similarité

Structures des données

- Matrice des données

$$\begin{bmatrix} x_{11} & \dots & x_{1f} & \dots & x_{1p} \\ \dots & \dots & \dots & \dots & \dots \\ x_{i1} & \dots & x_{if} & \dots & x_{ip} \\ \dots & \dots & \dots & \dots & \dots \\ x_{n1} & \dots & x_{nf} & \dots & x_{np} \end{bmatrix}$$

- Matrice de distances

$$\begin{bmatrix} 0 \\ d(2,1) & 0 \\ d(3,1) & d(3,2) & 0 \\ \vdots & \vdots & \vdots \\ d(n,1) & d(n,2) & \dots & \dots & 0 \end{bmatrix}$$

Notions de distance

- Définition :

Soit x et y deux vecteurs, d est une distance si et seulement si $d(x,y)$ vérifie les propriétés suivantes

- $d(x,y) \geq 0$
- $d(x,y) = 0 \Leftrightarrow x=y$
- $d(x,y) = d(y,x)$
- $d(x,y) \leq d(x,z) + d(z,y)$

Mesure de la qualité du clustering

- Mesure de Dissemblance/Ressemblance : la ressemblance est exprimée par une fonction de similarité $d(x, y)$.
- Il est difficile de définir “assez ressemblant” ou “bonne ressemblance” pour inclure deux individus dans le même groupe : Il y a typiquement une ***grande part de subjectivité*** dans la décision.

Dissemblance et ressemblance entre objets ou individus à valeurs réelles

- Une fonction de distance est normalement utilisée pour mesurer la ressemblance ou dissemblance entre deux individus.
- Une fonction de distance parmi les plus populaires pour des variables de type intervalle: *Minkowski distance* :

$$d(x, y) = \sqrt[q]{(|x_1 - y_1|^q + |x_2 - y_2|^q + \dots + |x_p - y_p|^q)}$$

où $x = (x_1, x_2, \dots, x_p)^T$ et $y = (y_1, y_2, \dots, y_p)^T$ sont deux vecteurs de dimension p représentant deux objets et q est un entier positif.

Dissemblance et ressemblance entre objets ou individus à valeurs réelles (suite)

- Si $q = 1$, $d(x,y)$ est la distance de Manhattan (ou distance l_1) :

$$d(x,y) = |x_1 - y_1| + |x_2 - y_2| + \dots + |x_p - y_p|$$

- Si $q = 2$, $d(x,y)$ est la distance Euclidienne :

$$d(x,y) = \sqrt{(|x_1 - y_1|^2 + |x_2 - y_2|^2 + \dots + |x_p - y_p|^2)}$$

- Si $q = \infty$, $d(x,y)$ n'est plus une distance mais correspond à une mesure très utilisée :

$$d(x,y) = \max(x, y)$$

Dissemblance et ressemblance entre objets ou individus à valeurs réelles (suite)

- Nous pouvons également utiliser une fonction de distance pondérée :

$$d(x, y) = \sqrt[q]{(w_1|x_1 - y_1|^q + w_2|x_2 - y_2|^q + \dots + w_p|x_p - y_p|^q)}$$

Par exemple, la distance de Mahalanobis d'une série de valeurs de moyenne $\mu = (\mu_1, \mu_2, \mu_3, \dots, \mu_p)^T$ et possédant une matrice de covariance Σ pour un vecteur à plusieurs variables $x = (x_1, x_2, x_3, \dots, x_p)^T$ est :

$$d(x) = \sqrt{(x - \mu)^T \Sigma^{-1} (x - \mu)}$$

Dissemblance et ressemblance entre objets ou individus à valeurs réelles (suite)

- La distance de Mahalanobis peut aussi être définie comme étant la mesure de dissimilarité entre deux vecteurs aléatoires et de **même distribution** avec une matrice de covariance Σ :

$$d(x, y) = \sqrt{(x - y)^T \Sigma^{-1} (x - y)}$$

- Si la matrice de covariance est la matrice identitaire, cette distance est alors la même que la distance euclidienne.
- Si la matrice de covariance est diagonale, elle est appelée distance euclidienne normalisée :

$$d(x, y) = \sqrt{\sum_{i=1}^p \frac{(x_i - y_i)^2}{\sigma_i^2}}$$

- A la différence de la distance euclidienne où toutes les composantes des vecteurs sont traitées de la même façon, la distance de Mahalanobis accorde un poids moins important aux composantes les plus bruitées (en supposant que chaque composante soit une variable aléatoire de type gaussien).

Dissemblance et ressemblance entre objets ou individus à valeurs réelles (suite)

- D'autres similarités et distances

- Similarité de corrélation: $S_{inner}(x, y) = x^T y$
- Similarité cosinus :

$$S_{\text{cosinus}}(x, y) = \arccos \frac{x^T y}{\|x\| \cdot \|y\|}$$

comme l'angle S_{cosinus} est compris dans l'intervalle $[0, \pi]$, la valeur π indiquera des vecteurs résolument opposés, $\pi/2$ des vecteurs indépendants (orthogonaux) et 0 des vecteurs colinéaires. Les valeurs intermédiaires permettent d'évaluer le degré de similarité (*cette métrique est fréquemment utilisée en fouille de textes*).

Exemple de problème de normalisation des données : cas des intervalles

Il faut standardiser les données en calculant une mesure normalisée par la moyenne et l'écart type du feature f : ce qu'on appelle le z-score.

Soit la matrice de n données suivantes :

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}$$

Exemple de problème de normalisation des données : cas des intervalles

- On définit alors la matrice standardisée des z-scores par :
 - Calculer l'écart absolu moyen:

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + \dots + |x_{nf} - m_f|)$$

où

$$m_f = \frac{1}{n}(x_{1f} + x_{2f} + \dots + x_{nf}).$$

- Calculer la mesure standardisée (z-score)

$$z_{if} = \frac{x_{if} - m_f}{s_f}$$

Exemple: distance de Manhattan

	Age	Salaire
Personne1	50	11000
Personne2	70	11100
Personne3	60	11122
Personne4	60	11074

$$d(p1, p2) = 120$$

$$d(p1, p3) = 132$$

Conclusion: p1 ressemble plus à p2 qu'à p3!!! 😞

	Age	Salaire
Personne1	-2	-0,5
Personne2	2	0,18
Personne3	0	0,32
Personne4	0	0

$$d(p1, p2) = 4,675$$

$$d(p1, p3) = 2,324$$

Conclusion: p1 ressemble plus à p3 qu'à p2 !!! 😊

$$m_{age} = 60, s_{age} = 5$$

$$m_{salaire} = 11074, s_{salaire} = 48$$

Dissemblance et ressemblance entre objets ou individus à valeurs discrètes

- Les coordonnées des vecteurs appartiennent à un ensemble fini $F = \{0, 1, \dots, k-1\}$, $k \geq 0$.
- Si $x, y \in F^p$, on définit la matrice de contingence $A(x,y)_{k \times k} = [a_{ij}]$ par $a_{ij} =$ le nombre de places où le premier vecteur a le symbole i et l'élément correspondant du second vecteur a le symbole j .

$$d_{\text{Hamming}}(x, y) = \sum_{i=0}^{k-1} \sum_{j=0, j \neq i}^{k-1} a_{ij}$$

Dissemblance et ressemblance entre objets ou individus à valeurs binaires

- Une table de contingence pour données binaires

		y		sum
		1	0	
x	1	a	b	$a+b$
	0	c	d	$c+d$
sum		$a+c$	$b+d$	p

a = nombre de positions où x à 1 et y à 1

- Exemple $x=(1,1,0,1,0)$ et $y=(1,0,0,0,1)$

$$a=1, b=2, c=1, d=2$$

Dissemblance et ressemblance entre objets ou individus à valeurs binaires

- Coefficient d'appariement (matching) simple (invariant pour variables symétriques):

$$S_{\text{appariement}}(x, y) = \frac{b + c}{a + b + c + d}$$

- Exemple $x=(1,1,0,1,0)$ et $y=(1,0,0,0,1)$

$$S_{\text{appariement}}(x, y) = 3/5$$

- Coefficient de Jaccard :

$$S_{\text{Jaccard}}(x, y) = 3/4$$

$$S_{\text{Jaccard}}(x, y) = \frac{b + c}{a + b + c}$$

Dissemblance et ressemblance entre objets ou individus à valeurs binaires

- L'indice de Tanimoto reprend l'idée de la similarité cosinus dans le cas des attributs discrets binaires :

$$S_{\text{Tanimoto}}(x, y) = \frac{x^T y}{\|x\|^2 + \|y\|^2 - x^T y}$$

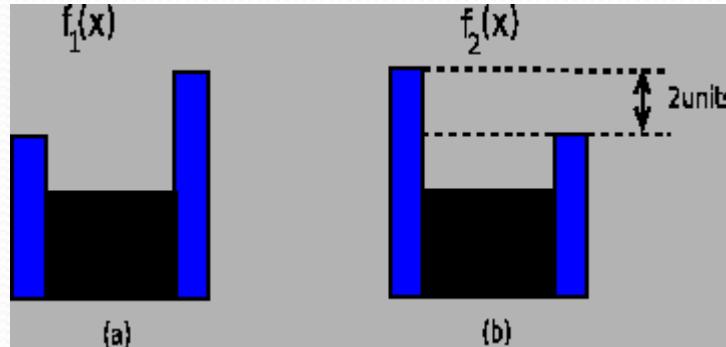
- La Earth Mover's Distance (*EMD, inventée par Rubin et al. en 1998!!*) a été définie comme “*the minimal cost that must be paid to transform one histogram (P) into the other (Q)*”:

$$EMD^D(P, Q) = (\min_{\{F_{ij}\}} \sum_{i,j} F_{ij} D_{ij}) / (\sum_{i,j} F_{ij}) \quad s.t. \quad F_{ij} \geq 0 \quad (1)$$

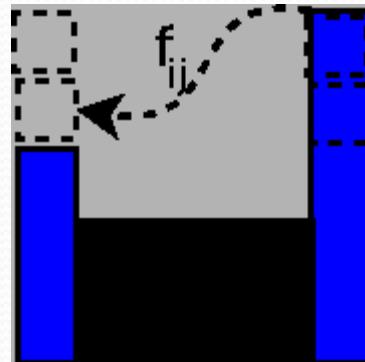
$$\sum_j F_{ij} \leq P_i, \quad \sum_i F_{ij} \leq Q_j, \quad \sum_{i,j} F_{ij} = \min(\sum_i P_i, \sum_j Q_j) \quad (2)$$

where $\{F_{ij}\}$ denotes the flows. Each F_{ij} represents the amount transported from the i^{th} supply to the j^{th} demand.

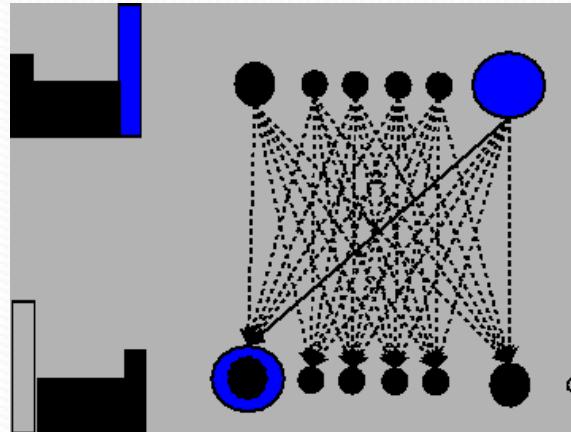
Dissemblance et ressemblance entre objets ou individus à valeurs binaires



- La *Earth Mover's Distance* c'est la quantité minimum d'effort requis pour transformer une distribution en une autre. Nous supposons que chaque distribution à la même moyenne dans notre cas :



Dissemblance et ressemblance entre objets ou individus à valeurs binaires



En 2008, Pele et Werman ont proposé une version modifiée de la distance EMD en étendant la version standard aux histogrammes non normalisés.

- Dans de très nombreuses situations de comparaison d'histogrammes, la différence entre valeurs fortes est moins important qu'entre valeurs faibles. La différence entre valeurs fortes devrait alors être réduite ou normalisée. La **distance Chi-2** est une distance entre histogrammes qui prend ça en compte :

$$\chi^2(P, Q) = \frac{1}{2} \sum_i \frac{(P_i - Q_i)^2}{(P_i + Q_i)}$$

Retour aux algorithmes de clustering

Méthodes de clustering : caractéristiques

- Extensibilité
- Abilité à traiter différents types de données
- Découverte de clusters de différentes formes
- Connaissances requises (paramètres de l'algorithme)
- Abilité à traiter les données bruitées et isolées

Algorithmes à partitionnement

- Construire une partition à k clusters d'une base D de n objets
- Les k clusters doivent optimiser le critère choisi
 - Global optimal : Considérer toutes les k -partitions
 - Heuristic methods : Algorithmes k -means et k -medoids
 - **k -means** (MacQueen'67) : Chaque cluster est représenté par son centre
 - **k -medoids or PAM** (Partition Around Medoids) (Kaufman & Rousseeuw'87) : Chaque cluster est représenté par un de ses objets

La méthode des k-moyennes (K-Means)

- L'algorithme k-means est en 4 étapes :
 1. Choisir k objets formant ainsi k clusters
 2. (Ré)affecter chaque objet O au cluster C_i de centre M_i tel que $\text{dist}(O, M_i)$ est minimal
 3. Recalculer M_i de chaque cluster (le barycentre)
 4. Aller à l'étape 2 si on vient de faire une affectation
- 

K-Means : Exemple

- $A = \{1, 2, 3, 6, 7, 8, 13, 15, 17\}$
- Créer 3 clusters à partir de A
- On prend 3 objets au hasard. Supposons que c'est 1, 2 et 3. Ca donne $C_1 = \{1\}$, $M_1 = 1$, $C_2 = \{2\}$, $M_2 = 2$, $C_3 = \{3\}$ et $M_3 = 3$
- Chaque objet O est affecté au cluster dont le milieu est le plus proche de O. 6 est affecté à C_3 car $\text{dist}(M_3, 6) < \text{dist}(M_2, 6)$ et $\text{dist}(M_3, 6) < \text{dist}(M_1, 6)$; On a
 - $C_1 = \{1\}$, $M_1 = 1$,
 - $C_2 = \{2\}$, $M_2 = 2$
 - $C_3 = \{3, 6, 7, 8, 13, 15, 17\}$, $M_3 = 69/7 = 9.86$

K-Means :Exemple (suite)

- $\text{dist}(3, M_2) < \text{dist}(3, M_3) \rightarrow 3$ passe dans C_2 . Tous les autres objets ne bougent pas.
 $C_1 = \{1\}$, $M_1 = 1$, $C_2 = \{2, 3\}$, $M_2 = 2.5$, $C_3 = \{6, 7, 8, 13, 15, 17\}$ et $M_3 = 66/6 = 11$
 - $\text{dist}(6, M_2) < \text{dist}(6, M_3) \rightarrow 6$ passe dans C_2 . Tous les autres objets ne bougent pas.
 $C_1 = \{1\}$, $M_1 = 1$, $C_2 = \{2, 3, 6\}$, $M_2 = 11/3 = 3.67$, $C_3 = \{7, 8, 13, 15, 17\}$, $M_3 = 12$
 - $\text{dist}(2, M_1) < \text{dist}(2, M_2) \rightarrow 2$ passe en C_1 . $\text{dist}(7, M_2) < \text{dist}(7, M_3) \rightarrow 7$ passe en C_2 . Les autres ne bougent pas. $C_1 = \{1, 2\}$, $M_1 = 1.5$, $C_2 = \{3, 6, 7\}$, $M_2 = 5.34$, $C_3 = \{8, 13, 15, 17\}$, $M_3 = 13.25$
 - $\text{dist}(3, M_1) < \text{dist}(3, M_2) \rightarrow 3$ passe en 1. $\text{dist}(8, M_2) < \text{dist}(8, M_3) \rightarrow 8$ passe en 2
 $C_1 = \{1, 2, 3\}$, $M_1 = 2$, $C_2 = \{6, 7, 8\}$, $M_2 = 7$, $C_3 = \{13, 15, 17\}$, $M_3 = 15$
- Plus rien ne bouge

Illustration (1)

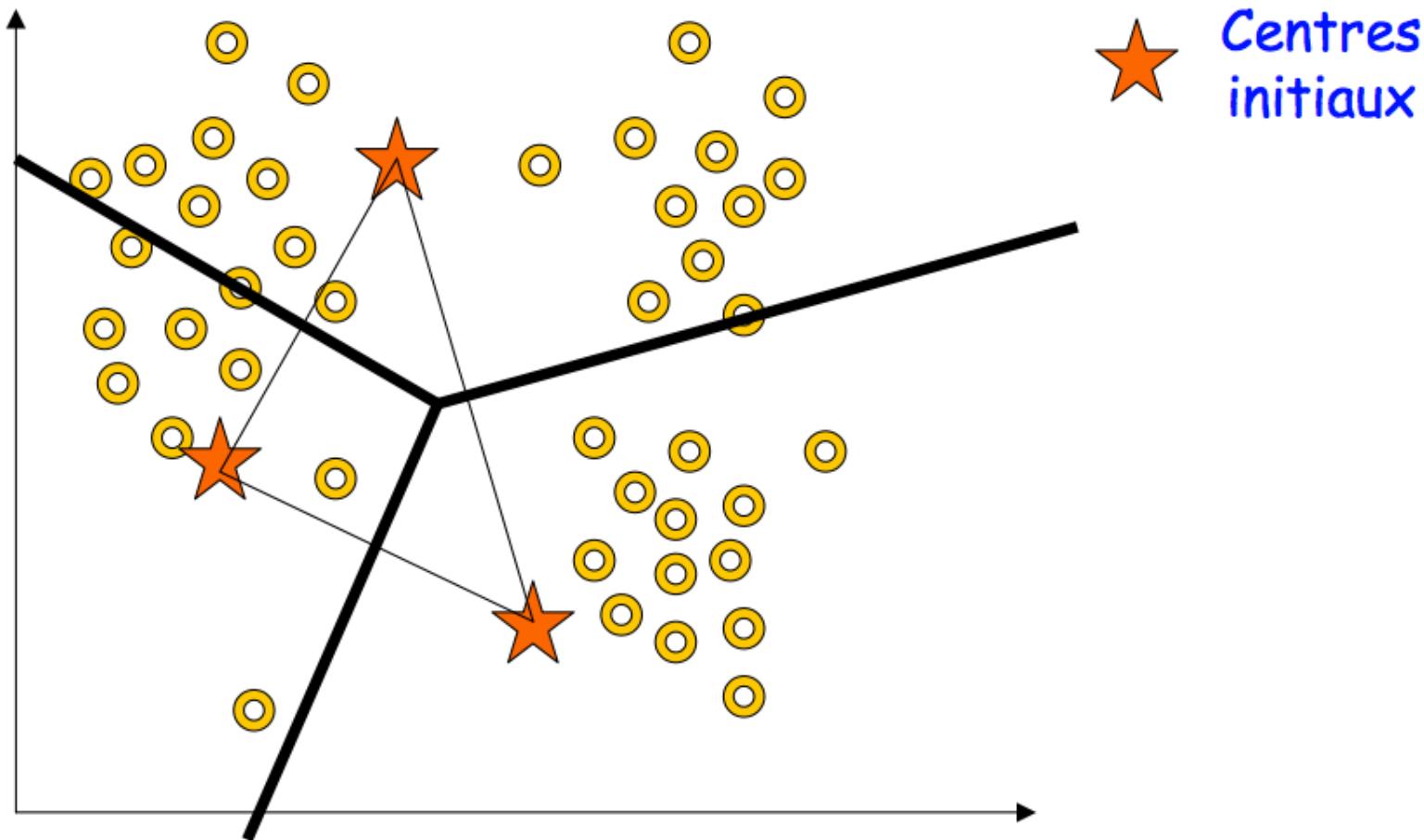


Illustration (2)

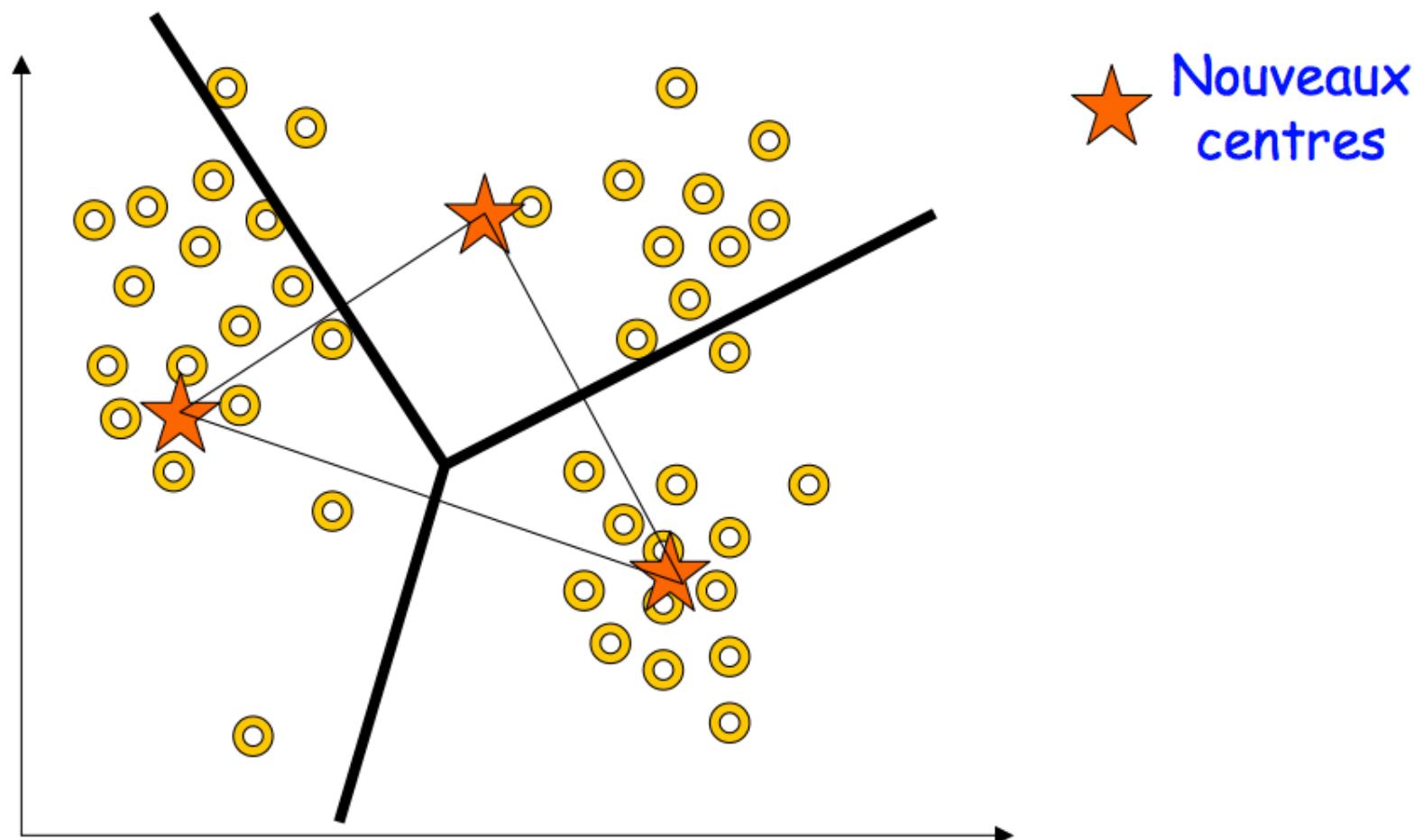
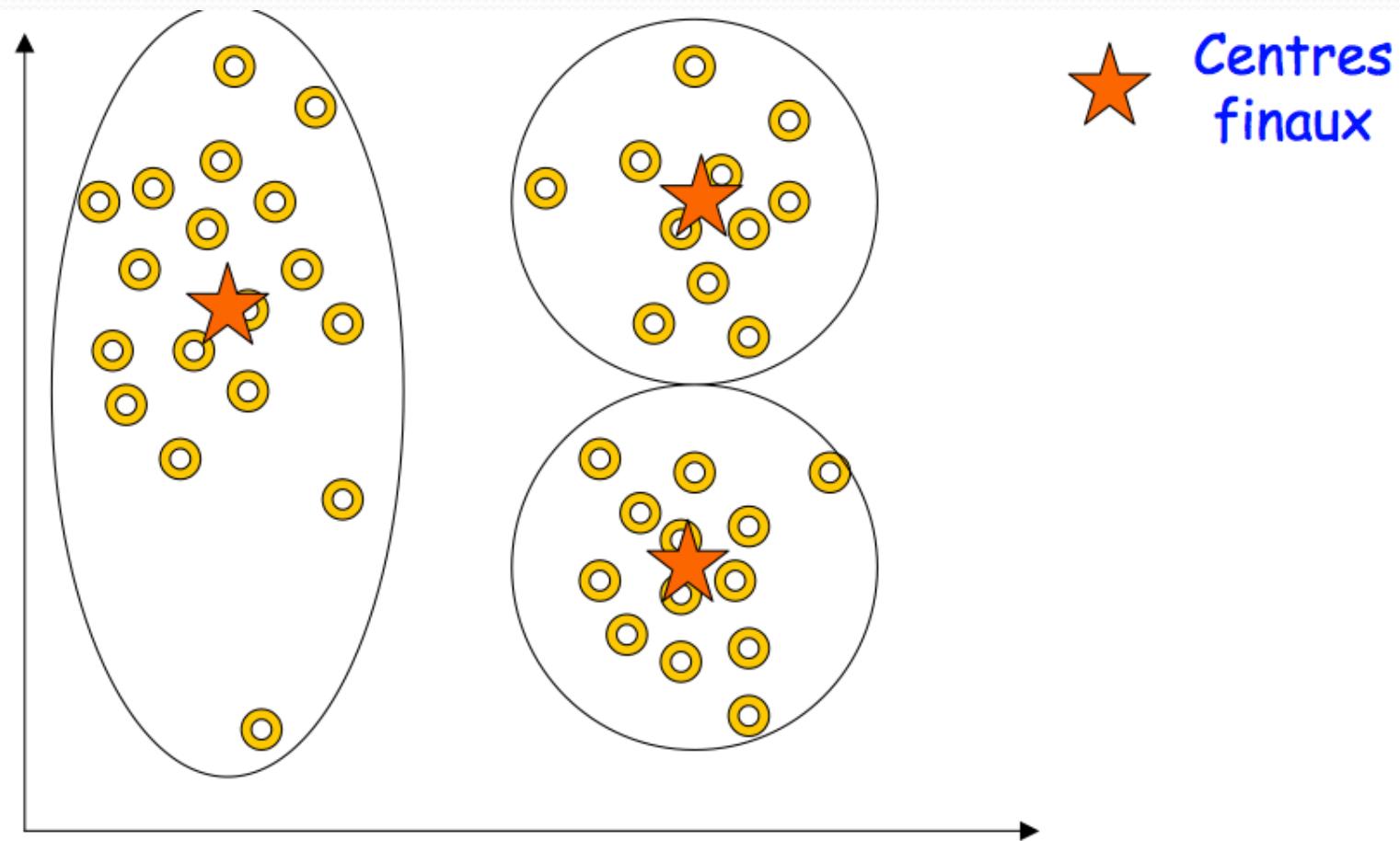


Illustration (3)



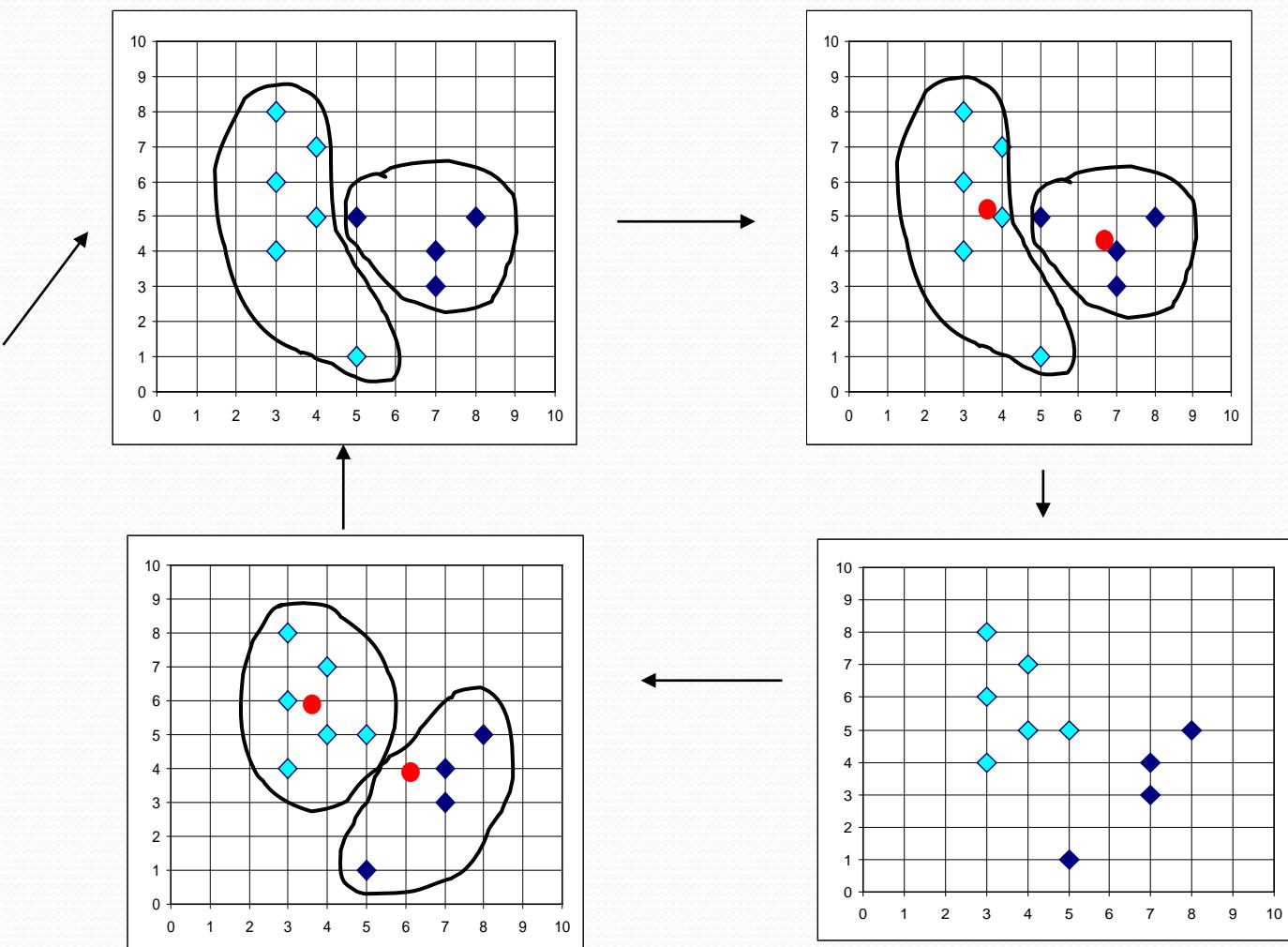
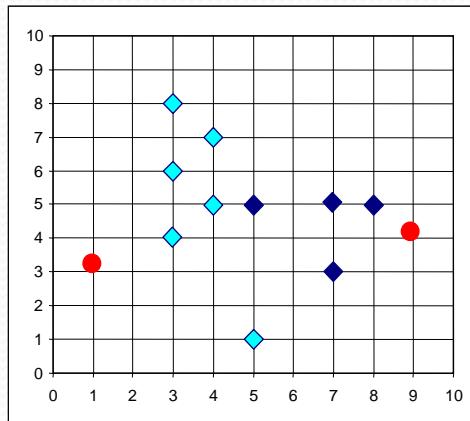
Exemple

- 8 points A, ..., H de l'espace euclidien 2D. k=2 (2 groupes)
- Tire aléatoirement 2 centres : B et D choisis.

points	Centre D(2,4), B(2,2)	Centre D(2,4), I(27/7,17/7)	Centre J(5/3,10/3), K(24/5,11/5)
A(1,3)	B	D	J
B(2,2)	B	I	J
C(2,3)	B	D	J
D(2,4)	D	D	J
E(4,2)	B	I	K
F(5,2)	B	I	K
G(6,2)	B	I	K
H(7,3)	B	I	K

Algorithme K-Means

- Exemple



Commentaires sur la méthode des K-Means

- Forces
 - **Relativement extensible** dans le traitement d'ensembles de taille importante
 - **Relativement efficace** : $O(t.k.n)$, où n représente # objets, k # clusters, et t # iterations. Normalement, $k, t \ll n$.
- Faiblesses
 - N'est pas applicable en présence d'attributs où la **moyenne** n'est pas définie (par exemple attribut nominal)
 - On doit spécifier **k** (nombre de clusters)
 - Incapable de traiter des données **bruitées**
 - Les clusters sont construits par rapports à des **objets inexistant**s (les milieux)
 - Ne peut pas découvrir les **groupes non-convexes**
 - Les **outliers** sont mal gérés.

La méthode des *K-Medoids* (PAM, Partitioning Around Medoids, 1987)

- Trouver des objets représentatifs (medoïdes) dans les clusters (au lieu de la moyenne)
- Principe
 - Commencer avec un ensemble de medoïdes puis itérativement remplacer un par un autre si ça permet de réduire la distance globale
 - Efficace pour des données de petite taille

Algorithme des k-Medoïdes

Choisir arbitrairement k medoides

Répéter

affecter chaque objet restant au medoïde le plus proche

Choisir aléatoirement un non-medoïde O_r

Pour chaque medoïde O_j ,

Calculer le coût TC du remplacement de O_j par O_r

Si $TC < 0$ alors

Remplacer O_j par O_r

Calculer les nouveaux clusters

Finsi

FinPour

Jusqu'à ce ce qu'il n'y ait plus de changement

Variantes des K-means

- Sélection des centres initiaux
- Calcul des similarités
- Calcul des centres (K-medoids : [Kaufman & Rousseeuw'87])
- GMM : Variantes de K-moyennes basées sur les probabilités
- K-modes : données catégorielles [Huang'98]
- K-prototype : données mixtes (numériques et catégorielles)

Aller plus loin avec K-means : ISODATA, C-Moyennes Floues

Isodata

- Intercalle des phases de fusion et d'éclatement de groupes dans l'algorithme "K-means".
 - fusion de 2 groupes si leur distance est faible
 - ex : distance inter-centre < seuil
 - éclatement d'un groupe en 2 sous-groupes si l'inertie du groupe est trop grande (> seuil)
 - paramètres (seuils) à fixer !!!

Notes sur l'algorithme ISODATA :

- Cet algorithme procédant par optimisation itérative est très simple et très connu.
- Il a des propriétés de convergence surprenantes *en pratique* (l'initialisation influe assez peu sur le résultat final par exemple) notamment pour traiter un grand nombre de données

C-moyennes floues (Fuzzy K-means)

- K-means est une classification dure :
 - Les objets sont membre d'une seule classe
- Fuzzy K-Means est une classification *floue* :
 - Les objets appartiennent à toutes les classes, à un degré variable d'appartenance.

Segmentation floue vs précise

Hard Clustering

Soit $X = \{x_1, \dots, x_N\}$

On appelle m -clustering de X la partition de X en m ensembles (clusters) C_1, \dots, C_m tels que :

$$C_i \neq \emptyset, i = 1, \dots, m$$

$$\bigcup_{i=1}^m C_i = X$$

$$C_i \cap C_j = \emptyset, i \neq j, i, j = 1, \dots, m$$

Soit $X = \{x_1, \dots, x_N\}$

Le m -clustering flou de X en m clusters est caractérisé par m fonctions d'appartenance u_j avec :

$$u_j : X \rightarrow [0,1], j = 1, \dots, m$$

$$\sum_{j=1}^m u_j(x_i) = 1, i = 1, \dots, N$$

$$0 < \sum_{i=1}^N u_j(x_i) < N, j = 1, \dots, m$$

Fuzzy
Clustering
Zadeh (1965)

Algorithmes hiérarchiques

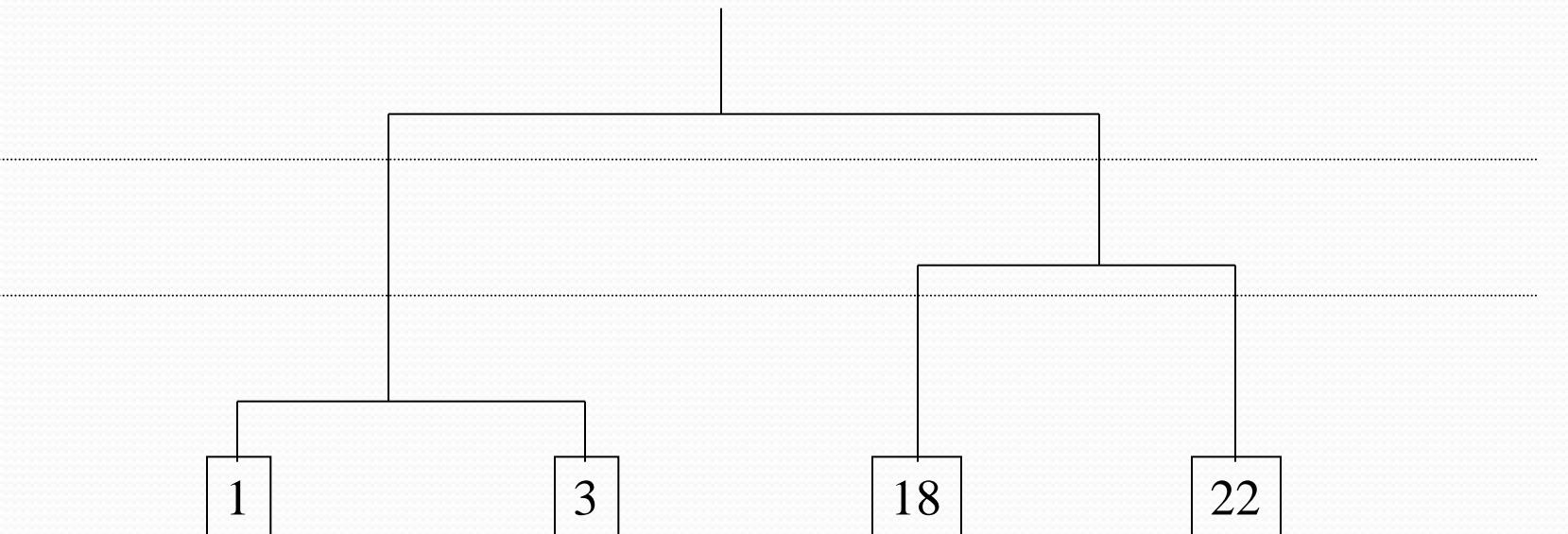
- **Une méthode hiérarchique** : construit une hiérarchie de clusters, non seulement une partition unique des objets.
- Le nombre de clusters k n'est pas exigé comme donnée
- Utilise une matrice de distances comme critère de clustering
- Une **condition de terminaison** peut être utilisée (ex. Nombre de clusters)

Agglomération - Principe

- Etapes :
 - Chaque individu représente un groupe
 - Trouver les deux groupes les plus proches
 - Grouper ces deux groupes en un nouveau groupe
 - Itérer jusqu'à N groupes

Agglomération

- Exemple



Variante

- Calculer les distances de tous les points deux à deux.
- Associer tous les points dont la distance ne dépasse pas un seuil.
- Calculer le centre de chaque cluster.
- Répéter le processus avec les centres et un nouveau seuil jusqu'à l'obtention du nombre de cluster souhaité.

Exemple (1)

- Points 27 - 51 - 52 - 33 - 45 - 22 - 28 - 44 - 40 - 38 - 20 - 57
- distance = $|p_1-p_2| / E(p_i)$; seuil = 10 %

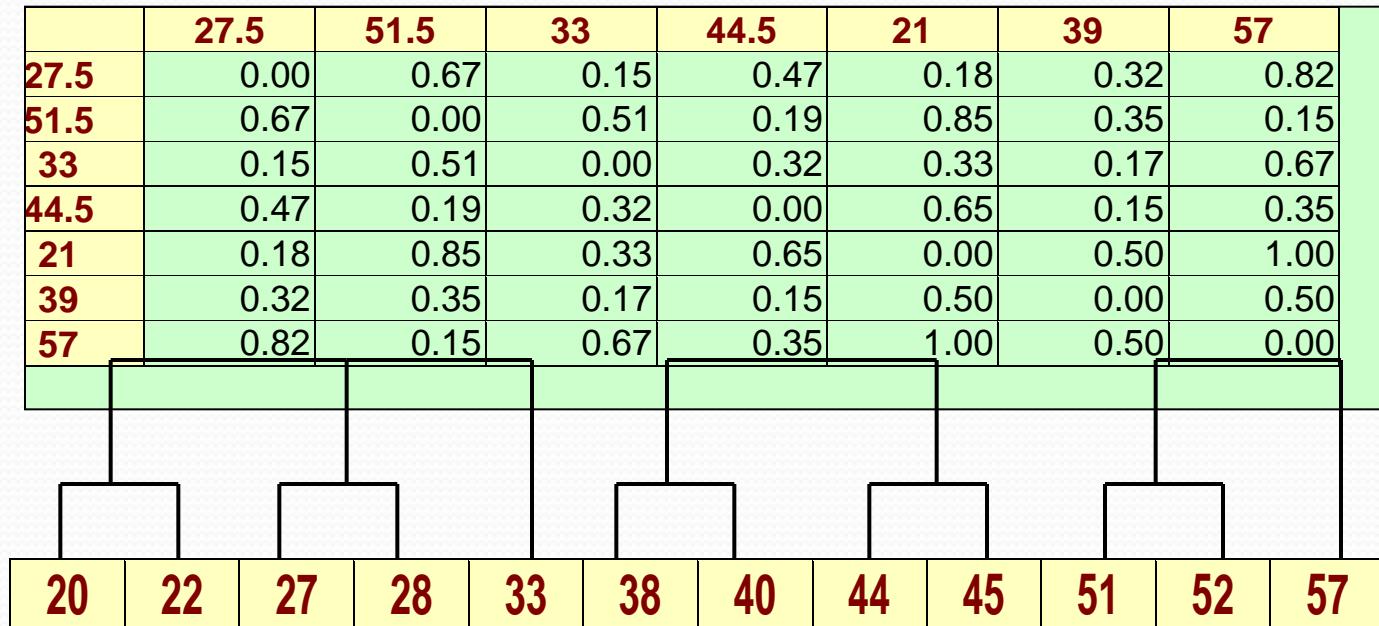
	27	51	52	33	45	22	28	44	40	38	20	57
27	0.00	0.65	0.68	0.16	0.49	0.14	0.03	0.46	0.35	0.30	0.19	0.81
51	0.65	0.00	0.03	0.49	0.16	0.78	0.62	0.19	0.30	0.35	0.84	0.16
52	0.68	0.03	0.00	0.51	0.19	0.81	0.65	0.22	0.32	0.38	0.86	0.14
33	0.16	0.49	0.51	0.00	0.32	0.30	0.14	0.30	0.19	0.14	0.35	0.65
45	0.49	0.16	0.19	0.32	0.00	0.62	0.46	0.03	0.14	0.19	0.68	0.32
22	0.14	0.78	0.81	0.30	0.62	0.00	0.16	0.59	0.49	0.43	0.05	0.95
28	0.03	0.62	0.65	0.14	0.46	0.16	0.00	0.43	0.32	0.27	0.22	0.78
44	0.46	0.19	0.22	0.30	0.03	0.59	0.43	0.00	0.11	0.16	0.65	0.35
40	0.35	0.30	0.32	0.19	0.14	0.49	0.32	0.11	0.00	0.05	0.54	0.46
38	0.30	0.35	0.38	0.14	0.19	0.43	0.27	0.16	0.05	0.00	0.49	0.51
20	0.19	0.84	0.86	0.35	0.68	0.05	0.22	0.65	0.54	0.49	0.00	1.00
57	0.81	0.16	0.14	0.65	0.32	0.95	0.78	0.35	0.46	0.51	1.00	0.00



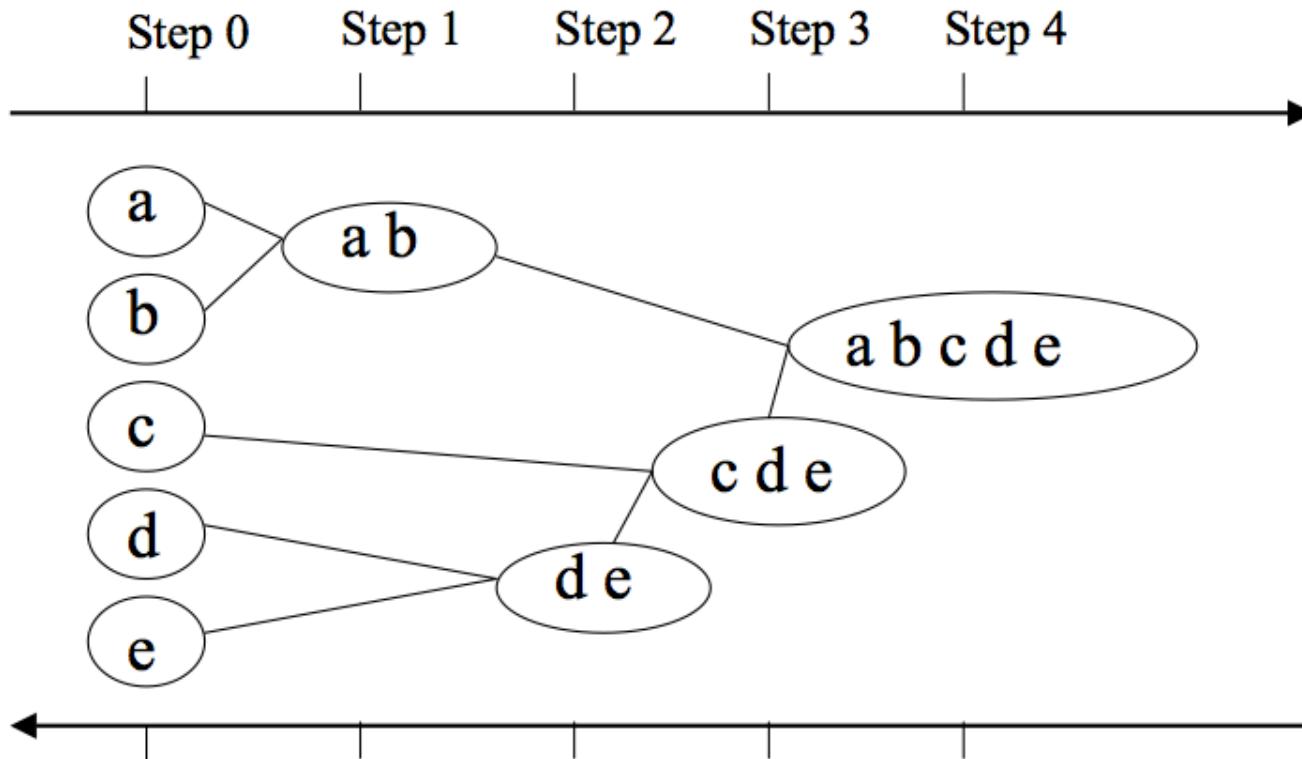
51	52	27	28	44	45	38	40	20	22	33	57
----	----	----	----	----	----	----	----	----	----	----	----

Exemple (2)

- Nouveaux centres
 - 27.5 - 51.5 - 33 - 44.5 - 21 - 39 - 57
- seuil = 20 %



Arbre de clusters



Dendrogramme

- Résultat : Graphe hiérarchique qui peut être coupé à un niveau de **dissimilitude** pour former une **partition**.
- La hiérarchie de clusters est représentée comme un arbre de clusters, appelé **dendrogramme**
- Les **feuilles** de l'arbre représentent les **objets**
- Les noeuds intermédiaires de l'arbre représentent les **clusters**

Distances entre clusters

- Distance entre les centres des clusters (Centroid Method) -> *faible résistance aux "outliers"*

- Distance minimale entre toutes les paires de données des 2 clusters (**Single Link Method**)

$$d(i, j) = \min_{x \in C_i, y \in C_j} \{d(x, y)\}$$

-> *tendance à former des chaînes*

- Distance maximale entre toutes les paires de données des 2 clusters (**Complete Link Method**)

$$d(i, j) = \max_{x \in C_i, y \in C_j} \{d(x, y)\}$$

-> *bonne méthode pour des grappes*

- Distance moyenne entre toutes la paires d'enregistrements (Average Linkage)

$$d(i, j) = \text{avg}_{x \in C_i, y \in C_j} \{d(x, y)\}$$

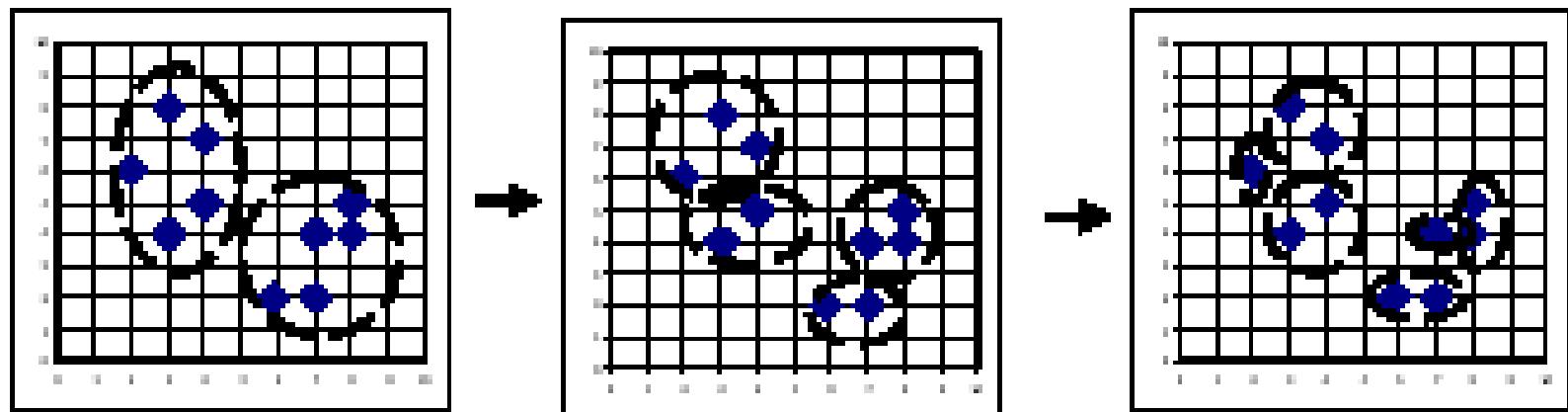
-> *plus difficile à calculer*

DIANA

- DIvide ANALysis
 - Kaufmann and Rousseeuw (1990)
- Méthode par division récursive
- Tous les objets sont placés dans une cluster
- Divise de manière hiérarchique les clusters
 - selon un critère de dispersion des objets
 - e.g., celle(s) dont les objets les plus proches sont les plus éloignés
- Stoppe quand le nombre de clusters est atteint ou les clusters contiennent 1 seul objet

Exemple

- Possibilité d'utiliser un seuil de distance

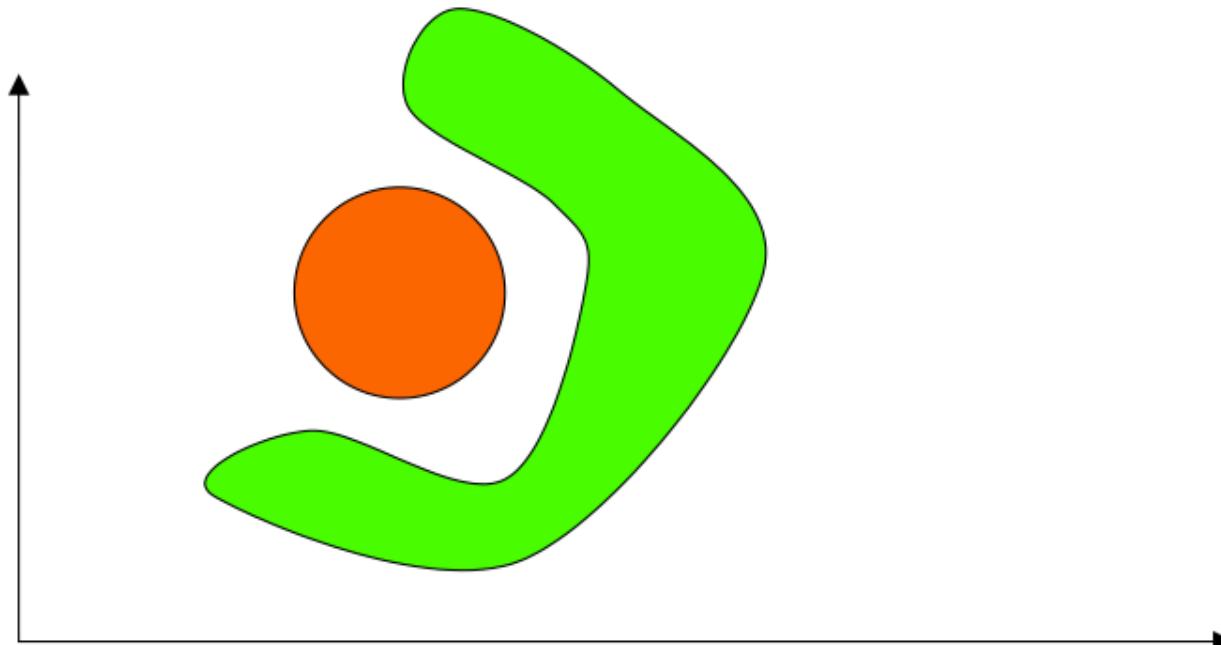


+ et - des algos hiérarchiques

- Avantages :
 - Conceptuellement simple
 - Propriétés théoriques sont bien connues
 - Quand les clusters sont groupés, la décision est définitive => le nombre d'alternatives différentes à examiner est réduit
- Inconvénients :
 - Groupement de clusters est définitif => décisions erronées sont impossibles à modifier ultérieurement
 - Méthodes non extensibles pour des ensembles de données de grandes tailles

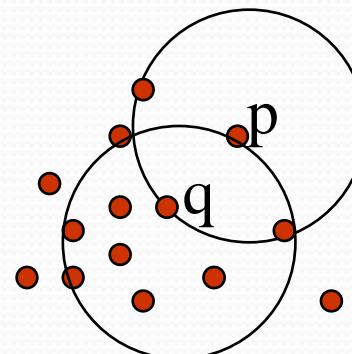
Méthode basée sur la densité

- Pour ce types de problèmes, l'utilisation de mesures de similarité (distance) est moins efficace que l'utilisation de **densité de voisinage**.



Clustering basé sur la densité

- Voit les clusters comme des régions denses séparées par des régions qui le sont moins (bruit)
- Deux paramètres:
 - Eps: Rayon maximum du voisinage
 - MinPts: Nombre minimum de points dans le voisinage-Eps d'un point
- Voisinage : $V_{Eps}(p)$: $\{q \in D \mid \text{dist}(p,q) \leq Eps\}$
- Un point p est directement densité-accessible à partir de q resp. à Eps, MinPts si
 - $p \in V_{Eps}(q)$
 - $|V_{Eps}(q)| \geq \text{MinPts}$

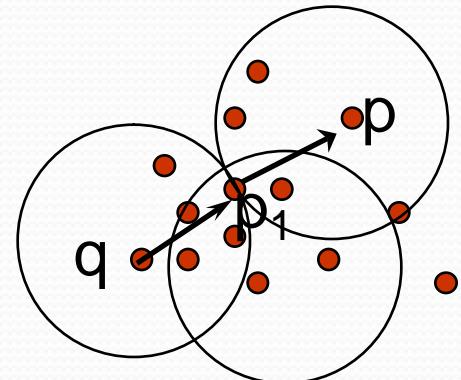


$\text{MinPts} = 5$
 $Eps = 1 \text{ cm}$

Clustering basé sur la densité

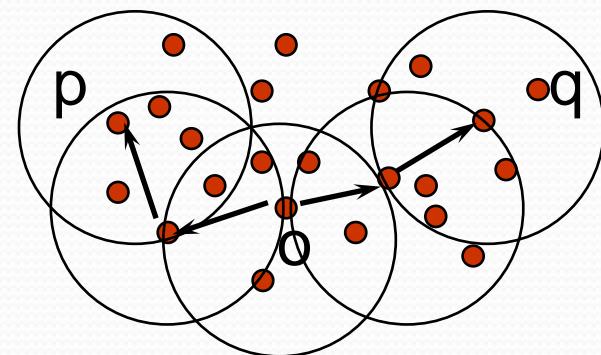
- Accessibilité:

- p est accessible à partir de q resp. à $Eps, MinPts$ s'il existe p_1, \dots, p_n , $p_1 = q$, $p_n = p$ t.q p_{i+1} est directement densité-accessible à partir de p_i



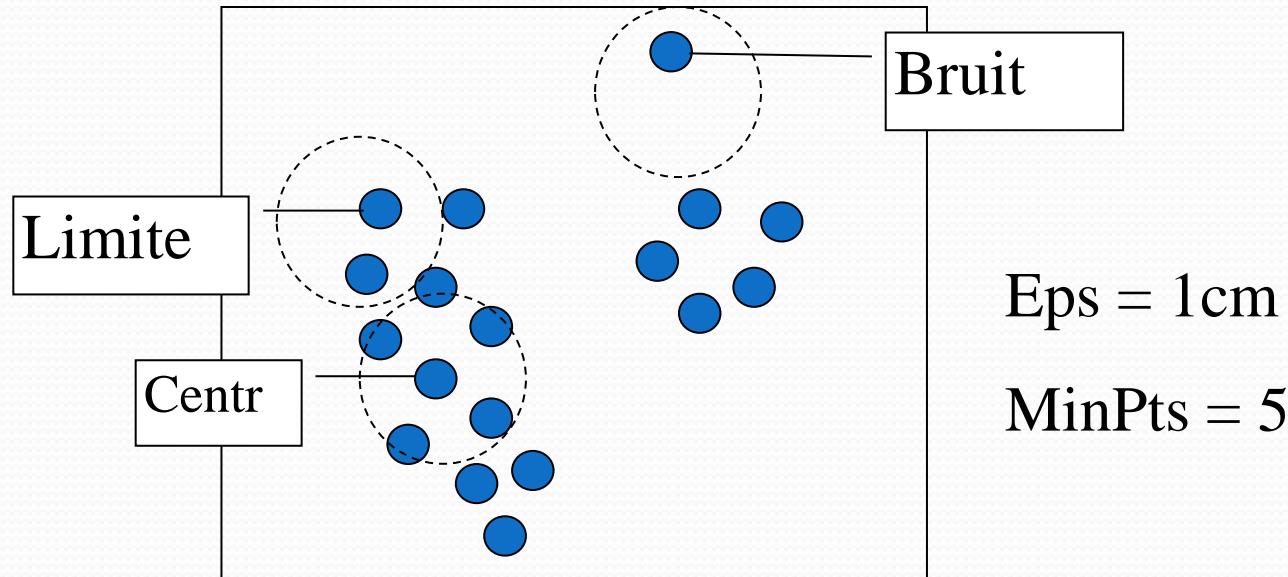
- Connexité

- p est connecté à q resp. à $Eps, MinPts$ s'il existe un point o t.q p et q accessibles à partir de o resp. à Eps et $MinPts$.



DBSCAN: Density Based Spatial Clustering of Applications with Noise

- Un cluster est l'ensemble maximal de points connectés
- Découvre des clusters non nécessairement convexes

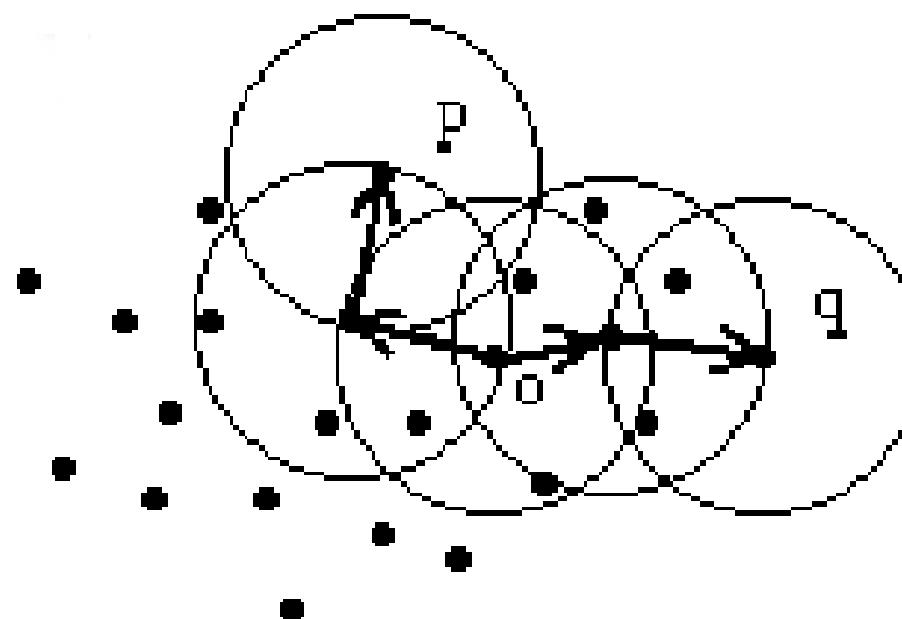


DBSCAN: l'algorithme

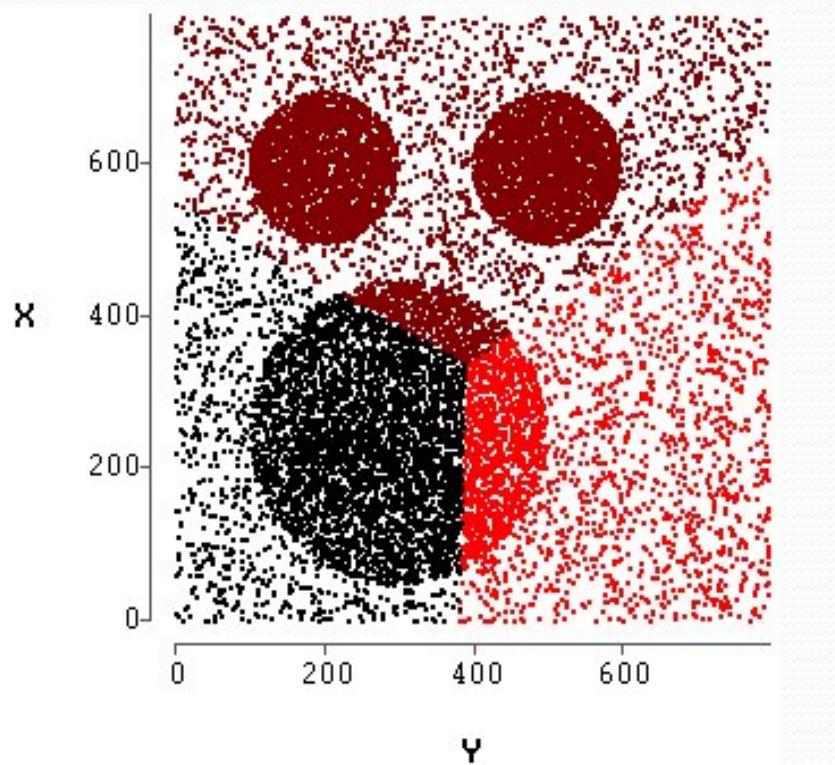
- Choisir p
- Récupérer tous les points accessibles à partir de p resp. à Eps et $MinPts$.
- Si p est un centre, un cluster est formé.
- si p est une limite, alors il n'y a pas de points accessibles de p : passer à un autre point
- Répéter le processus jusqu'à épuiser tous les points.
- *Performance de DBSCAN $O(n \log n)$*

Exemple

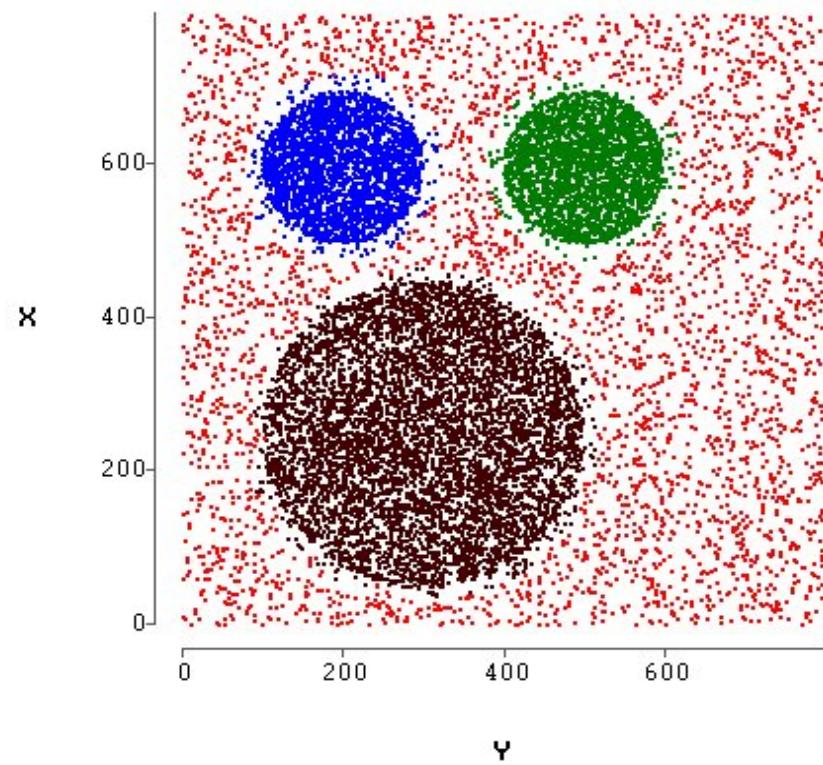
- on commence par o, ce cluster contient p, q, etc.



Comparaison K-means et DBSCAN



Résultat de K-means



Résultat de Density

Résumé

- Le clustering groupe des objets en se basant sur leurs **similarités**.
- Le clustering possède plusieurs applications.
- La mesure de similarité peut être calculée pour **differents types** de données.
- La sélection de la **mesure de similarité** dépend des données utilisées et le type de similarité recherchée.

- Les méthodes de clustering peuvent être classées en :
 - Méthodes de partitionnement,
 - Méthodes hiérarchiques,
 - Méthodes à densité de voisinage
- Plusieurs travaux de recherche sur le clustering en cours...
- Plusieurs applications en **perspective** : Génomique, Environnement, ...

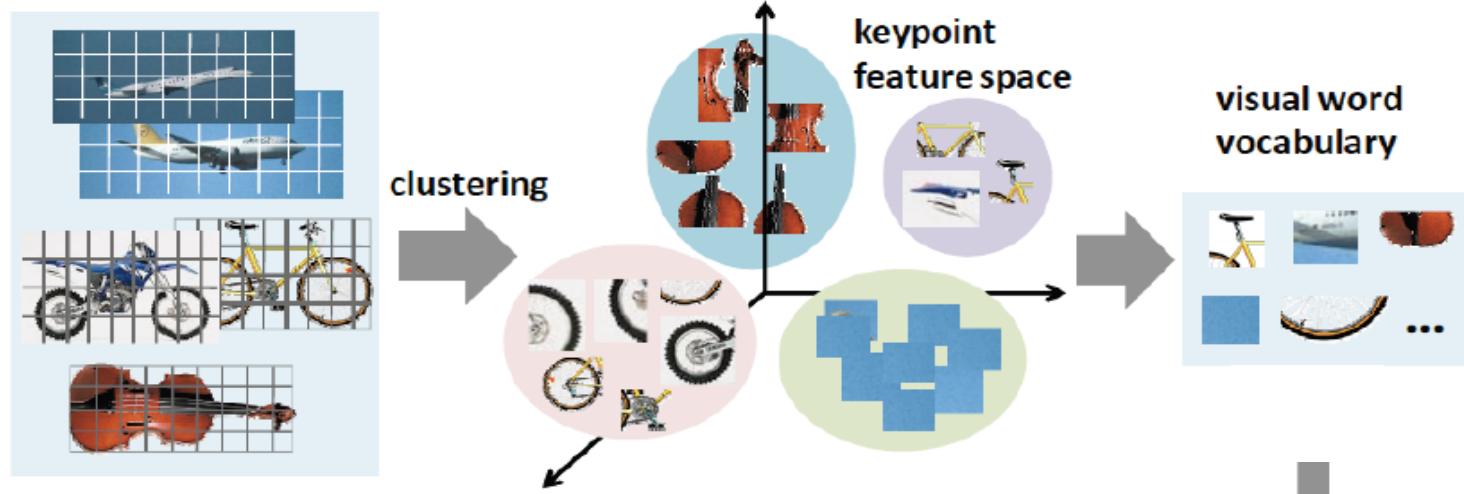
- Recherches en cours
 - Passage à l'échelle
 - échantillonnage, optimisation, indexation, mixages, ...
 - Prise en compte de contraintes
 - connaissances utilisateur ou données
 - objectifs utilisateur

Bag of Visual Words : partie offline

➤ Extraction descripteurs, approches basées dictionnaire « visuel »

➤ K-Means

- Limitations : concentre les clusters sur les zones de fortes densités de points, délaisse les autres
- Alternatives : méthodes à bases radiales, mais inconvénients inverses
- En pratique le K-Means reste couramment utilisé
- Apprentissage de dictionnaire (non supervisé ou supervisé)

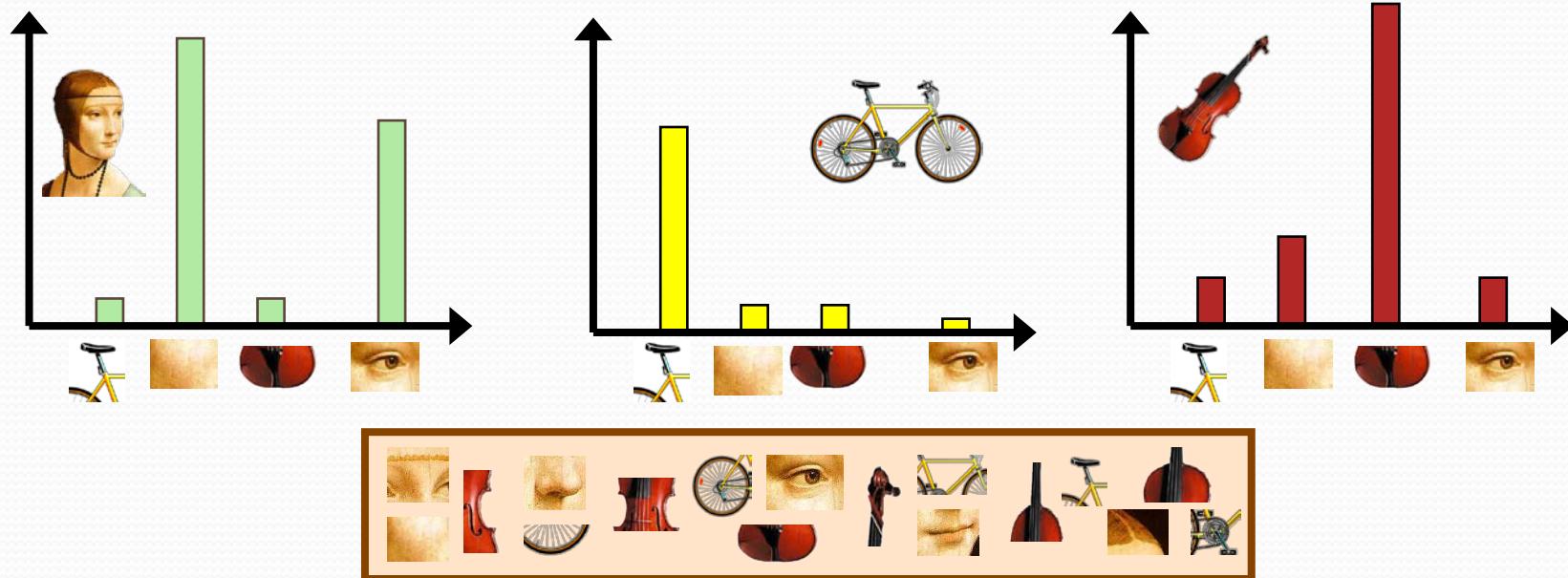


Bag of Visual Words : partie online

2 étapes :

- Codage : projection chaque descripteur local => dictionnaire
- Agrégation projections : résumé statistique / mot visuel

=> Index global image : vraisemblance contenir chaque mot visuel



Coding

Notations :

- Les données images : $\mathbf{X} = \left\{ x_j \in \mathbb{R}^d \right\}, j \in \{1; N\}$
 - Les centres : $\mathbf{C} = \{C_m\}, m \in \{1; M\}$
 - Coding :

$$\begin{aligned}
 & f: \mathbb{R}^d \rightarrow \mathbb{R}^M \\
 & x_j \rightarrow f(x_j) = \alpha_j = \{\alpha_{m,j}\}, \quad m \in \{1; M\} \\
 & \text{H} = c_1 \begin{bmatrix} \alpha_{1,1} & \cdots & \alpha_{1,j} & \cdots & \alpha_{1,N} \\ \vdots & & \vdots & & \vdots \\ \alpha_{m,1} & \cdots & \alpha_{m,j} & \cdots & \alpha_{m,N} \\ \vdots & & \vdots & & \vdots \\ \alpha_{M,1} & \cdots & \alpha_{M,j} & \cdots & \alpha_{M,N} \end{bmatrix} \Rightarrow g: \text{pooling} \\
 & \qquad \qquad \qquad \Downarrow \\
 & \qquad \qquad \qquad f: \text{cooding}
 \end{aligned}$$

Coding

- Hard assignment : le plus simple (cf text retrieval)

Hard coding: $f = f_Q$ assigns a constant weight to its closest center:

$$f_Q(x_j)[m] = \begin{cases} 1 & \text{if } m = \underset{k \in \{1;M\}}{\operatorname{argmin}} \|x_j - c_k\|^2 \\ 0 & \text{otherwise} \end{cases}$$

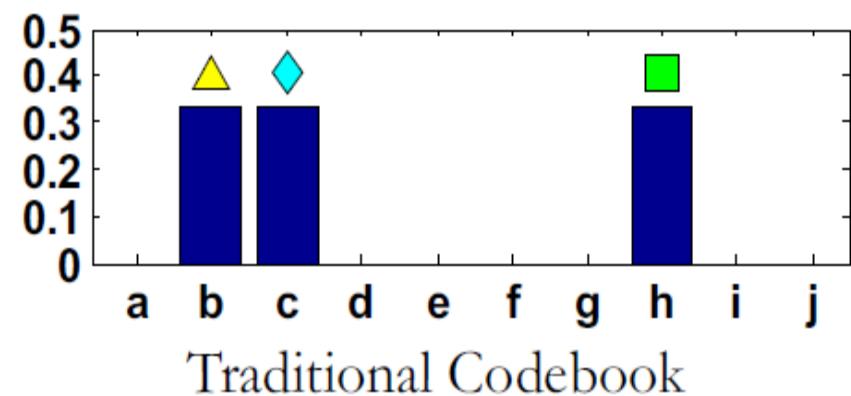
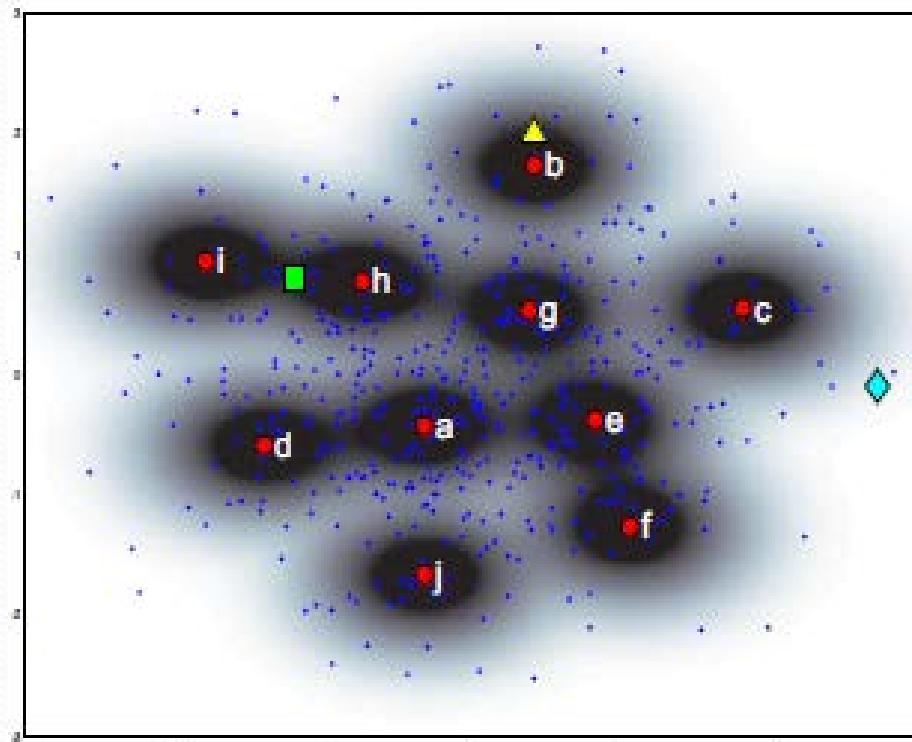
- Pour etre plus précis

- Soft assignment
- Sparse coding
- Importance de la localité dans le codage

=> Index global image : vraisemblance contenir chaque mot visuel

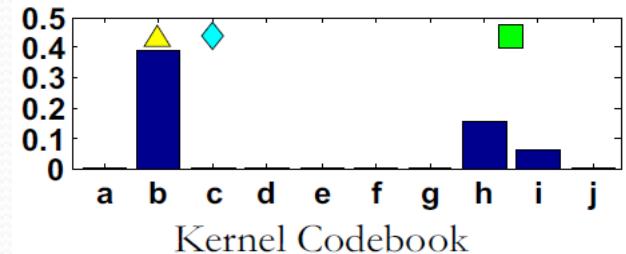
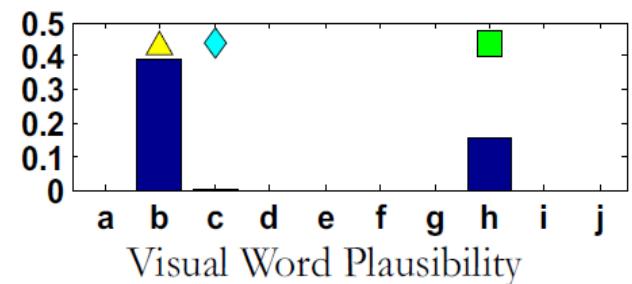
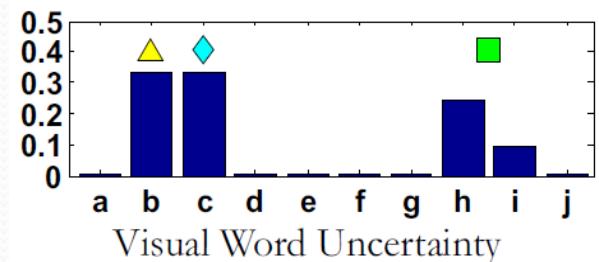
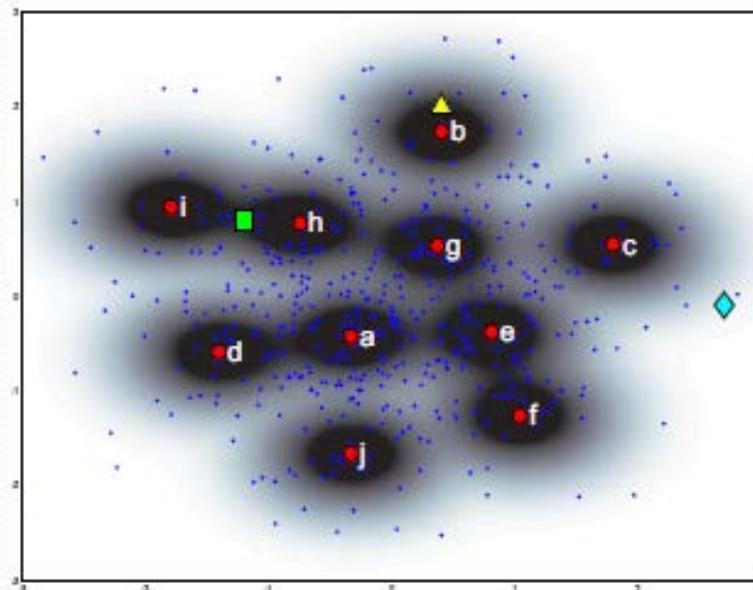
Projection signature locale => dictionnaire

- Idée la plus simple : hard assignement
 - On cherche le cluster le plus proche du descripteur
 - On lui assigne un poids fixe (*e.g.* 1)



Projection signature locale => dictionnaire

- Plus sophistiqué : soft assignment
 - Kernel codebook : poids absolu
 - Uncertainty : poids relatif
 - Plausibility : poids absolu au 1-nn



Projection signature locale => dictionnaire

- **Soft vs hard assignement**
 - En pratique, le gain du soft / hard n'est pas toujours clair
 - Seul la stratégie basée Uncertainty améliore les performances de classification
- **Autre approche : sparse coding**
 - On approxime chaque descripteur local comme une combinaison linéaire d'un sous-ensemble de mots du dictionnaire $x_i = D \alpha$

Projection signature locale => dictionnaire

- **Sparse coding**
 - On approxime chaque descripteur local comme une combinaison linéaire d'un sous-ensemble de mots du dictionnaire : $x_i = D \alpha$
 - α poids, D dictionnaire
 - On force chaque x_i à n'être représenté que par un petit nombre de mots visuel => parcimonie

$$\alpha_i = \underset{\alpha}{\operatorname{argmin}} L(\alpha, D) \triangleq \|x_i - D\alpha\|_2^2 + \lambda \|\alpha\|_1$$

Projection signature locale => dictionnaire

- **Sparse coding vs VQ : hard assignement classique**
 - SC : Sparse Coding : la majorité des $\alpha_i = 0$
 - LLC : Local LinearCoding : Les mots représentants doivent être proches (localité)
 - Question : ouverte : pertinence de ces critères de minimisation de l'erreur de reconstruction en classification ?

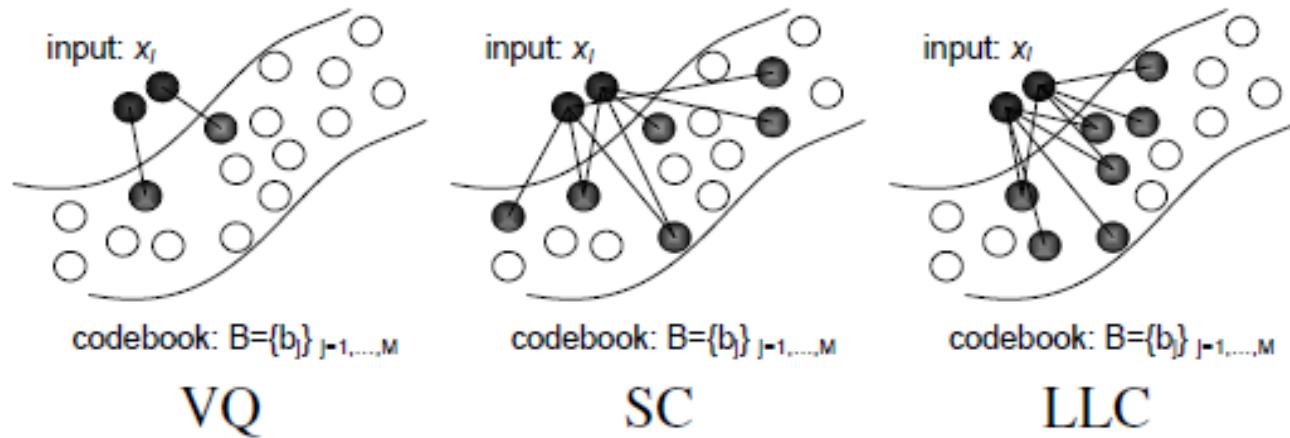


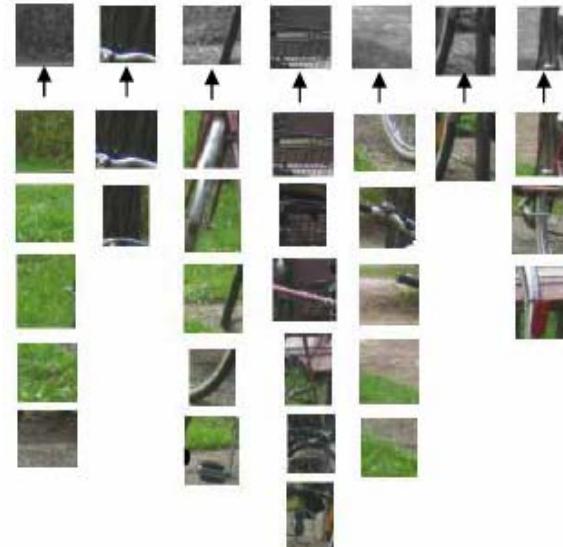
Figure 2. Comparison between VQ, SC and LLC. *The selected bases for representation are highlighted in red*

Pooling : Agglomération projections => index global image

$$\mathbf{H} = \begin{matrix} & x_1 & x_j & x_N \\ c_1 & \left[\begin{matrix} \alpha_{1,1} & \cdots & \alpha_{1,j} & \cdots & \alpha_{1,N} \\ \vdots & & \vdots & & \vdots \\ \alpha_{m,1} & \cdots & \alpha_{m,j} & \cdots & \alpha_{m,N} \\ \vdots & & \vdots & & \vdots \\ \alpha_{M,1} & \cdots & \alpha_{M,j} & \cdots & \alpha_{M,N} \end{matrix} \right] & c_m \\ & & & \Rightarrow g: \text{pooling} \\ c_M & & & \\ & & \Downarrow & \\ & & f: \text{cooding} & \end{matrix}$$

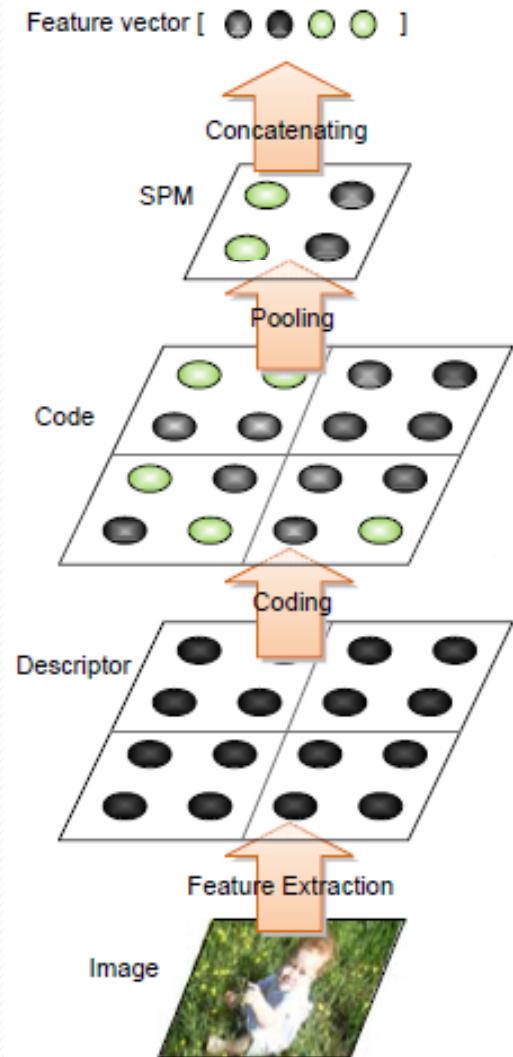
Agglomération projections => index global image

- Index global : vraisemblance image de contenir chaque mot visuel
- Différentes manières de résumer les projections
 - Sum pooling : approche BoW classique
 - Issue recherche textuelle : compter les occurrences de mots dans le document



Agglomération projections => index global image

- Index global : vraisemblance image de contenir chaque mot visuel
- Différentes manières de résumer les projections
 - Max pooling: on garde la valeur max de la projection pour chaque mot visuel
 - Intéressant pour du sparse / soft coding : limite l'impact du bruit
 - En lien avec les modèles d'inspiration biologique (cortex)



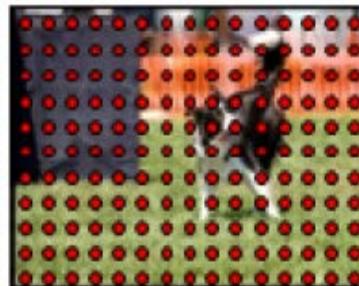
Agglomération projections => index global image

- BoW : histogramme : toute l'information spatiale sur la localisation des descripteurs locaux est perdu
 - Comment conserver cette information ??
- Mettre en place l'approche BoW dans différentes sous-régions de l'image
 - Concaténation des BoW ?
 - Avoir une mesure de similarité combinant les niveaux
 - Spatial Pyramid Matching [LAZEBNIK06]
 - Extension au domaine spatial du pyramid match kernel[GRAUMAN05]

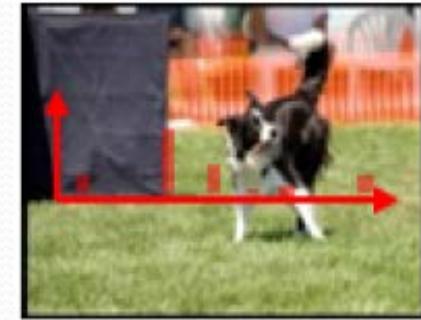
Agglomération projections => index global image



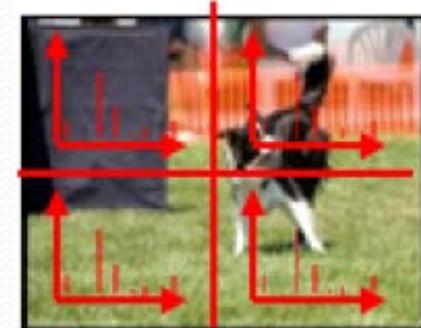
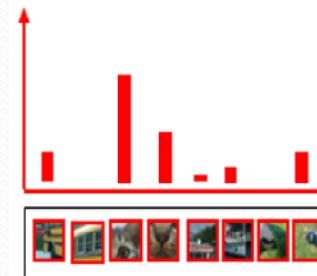
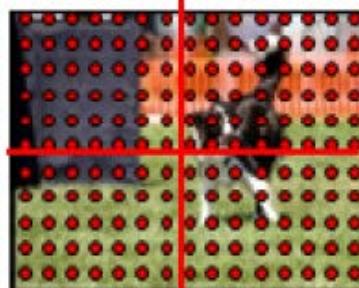
1x1



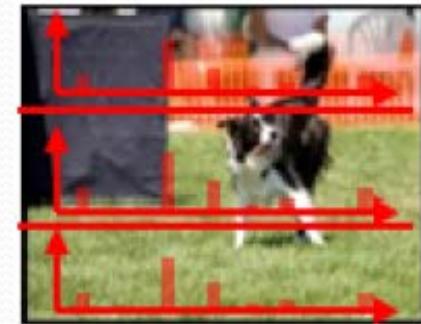
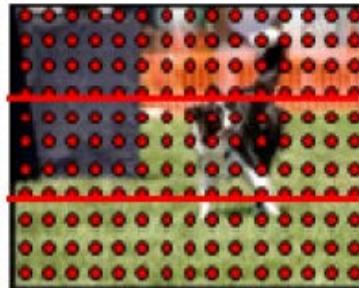
Vocabulary Assignment
(Bag-of-Words model)



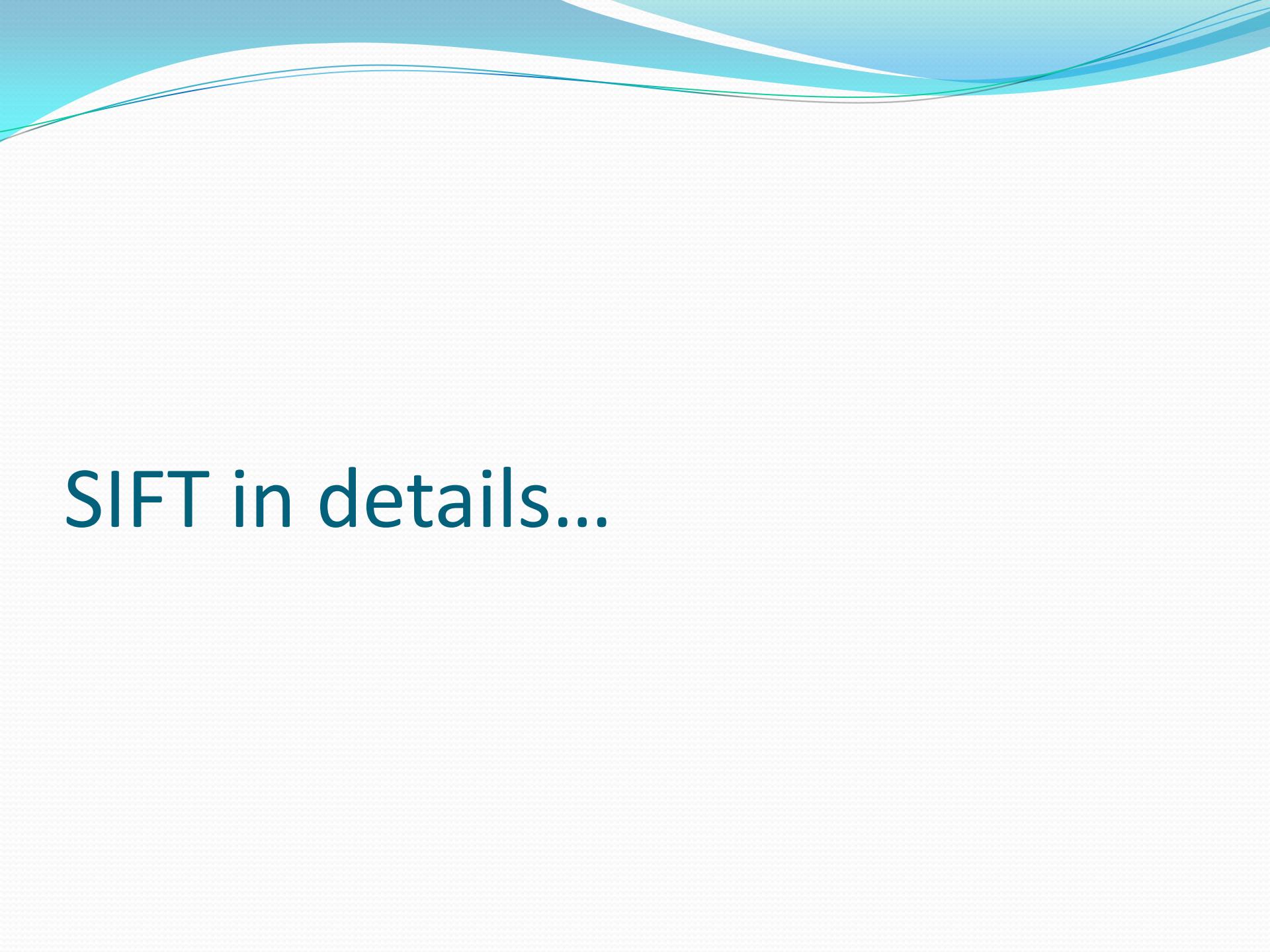
2x2



3x1



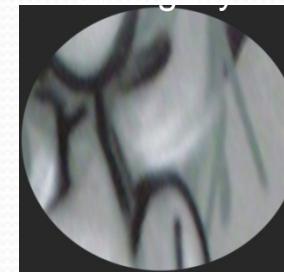
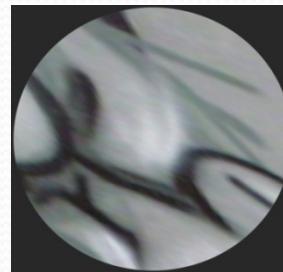
Reminder on SIFT & SURF



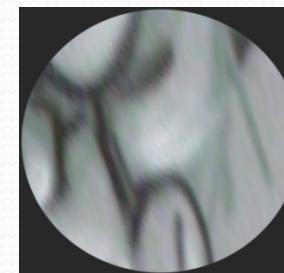
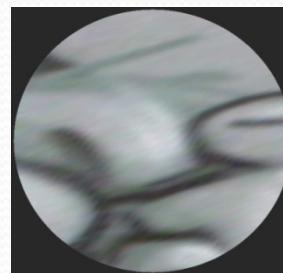
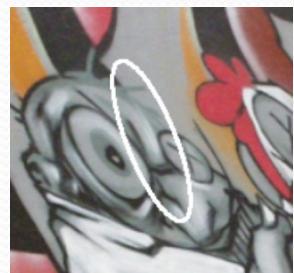
SIFT in details...

Region Detection Steps: Review

Feature
Extraction



*	*	*	*
*	*	*	*
*	*	*	*
*	*	*	*



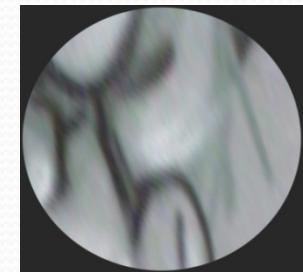
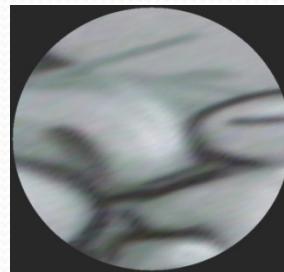
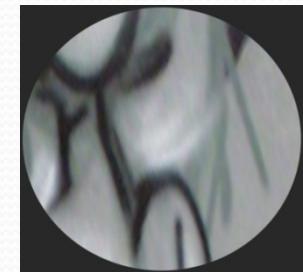
*	*	*	*
*	*	*	*
*	*	*	*
*	*	*	*

Scale Invariant Feature Transform (SIFT)

- Remember how we resolved the orientation ambiguity?

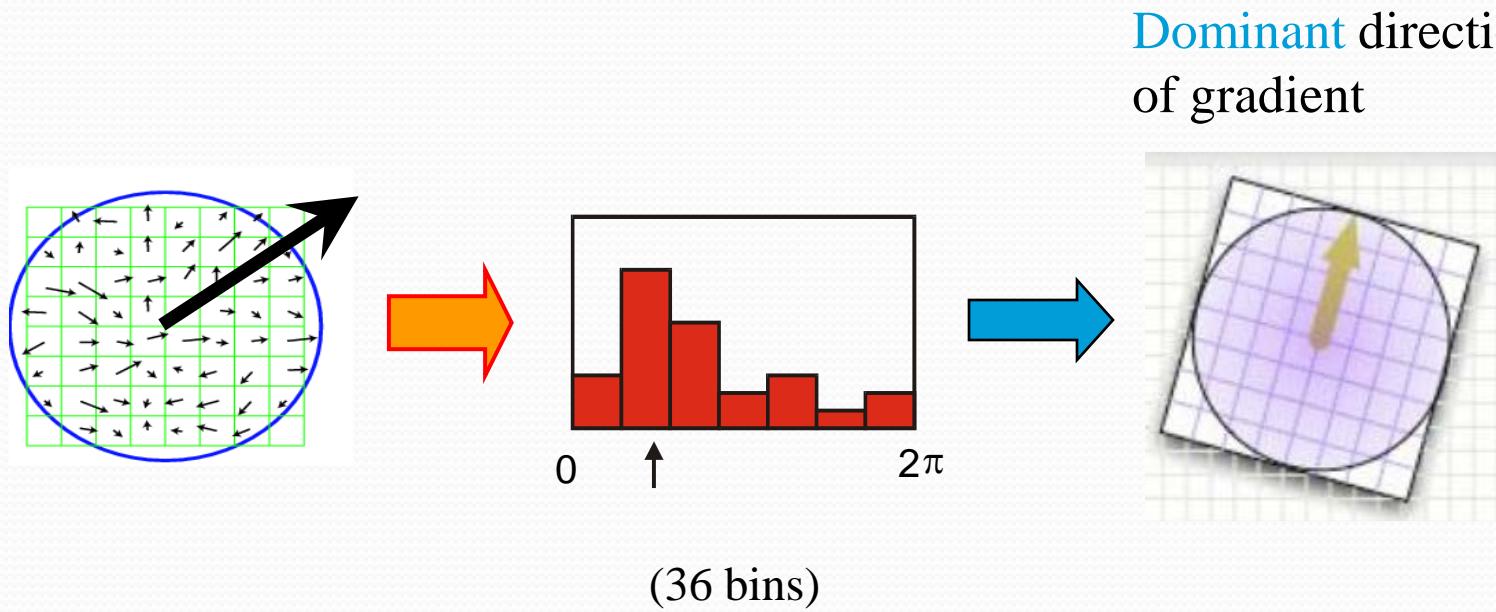


?



Scale Invariant Feature Transform (SIFT)

- Find dominant gradient direction using the histograms of gradient direction.

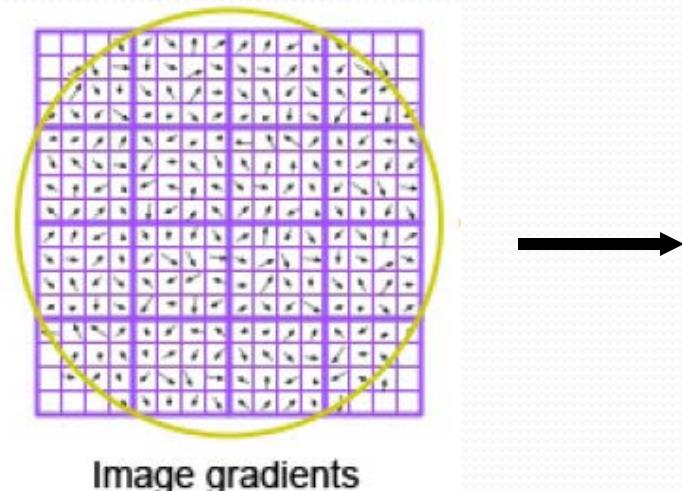


Scale Invariant Feature Transform (SIFT)

- Same theory, except that we use 16 histograms (8 bins each).

Main idea:

1. Take a 16×16 window around detected interest point
2. Divide into a 4×4 grid of cells
3. Compute histogram in each cell



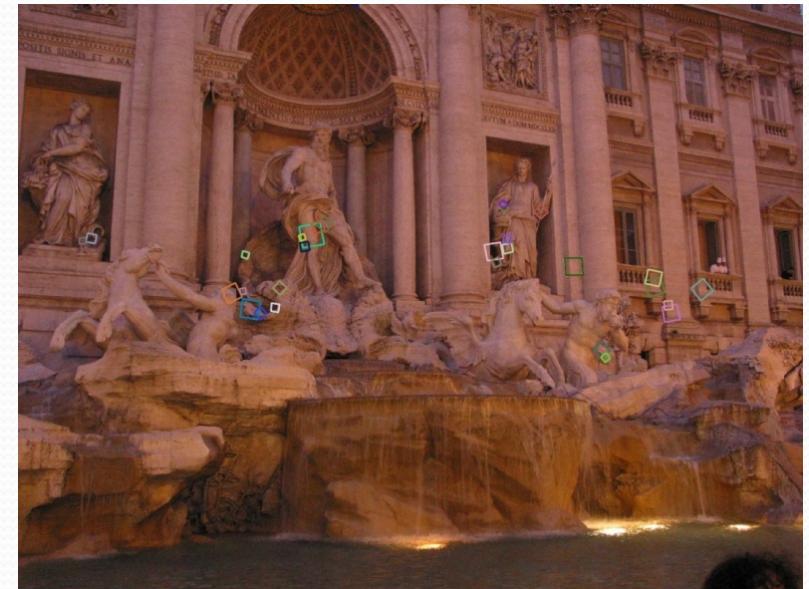
16 histograms x 8 orientations
= 128 features

Properties of SIFT

- Highly distinctive!
 - A single feature can be correctly matched with high probability against a large database of features from many images.
- Scale and rotation invariant.
- Partially invariant to 3D camera viewpoint
 - Can tolerate up to about 60 degree out of plane rotation
- Can be computed fast and efficiently

Properties of SIFT (cont'd)

- Partially invariant to changes in illumination



http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT

SIFT – Main Steps

(1) Scale-space extrema detection

- Extract scale and rotation invariant interest points (i.e., keypoints).

(2) Keypoint localization

- Determine location and scale for each interest point.

(3) Orientation assignment

- Assign one or more orientations to each keypoint.

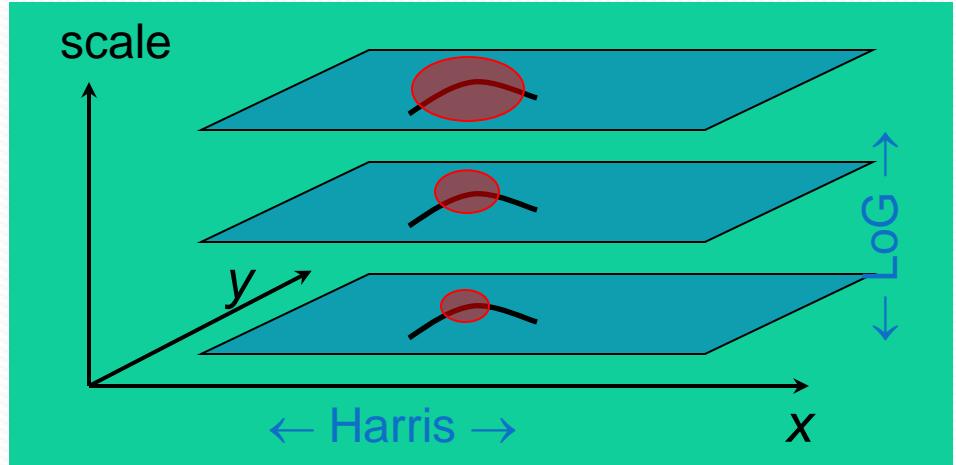
(4) Keypoint descriptor

- Use local image gradients at the selected scale.

D. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints”, **International Journal of Computer Vision**, 60(2):91-110, 2004. Cited 9589 times (as of 3/7/2011)

1. Scale-space Extrema Detection

- Harris-Laplacian
- *Find local maxima of:*
 - Harris detector in space
 - LoG in scale

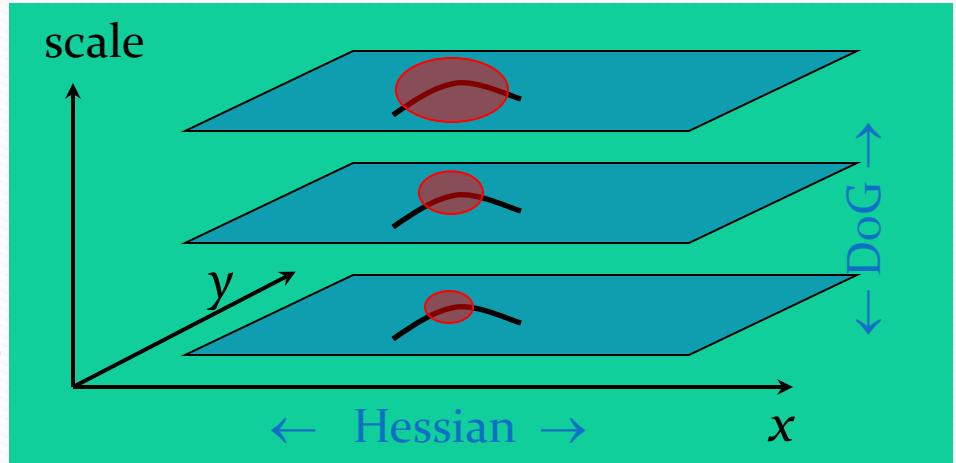


- SIFT

Find local maxima of:
- DoG in space
- DoG in scale

$$\sigma_n = k^n \sigma_0$$

($k=2$)



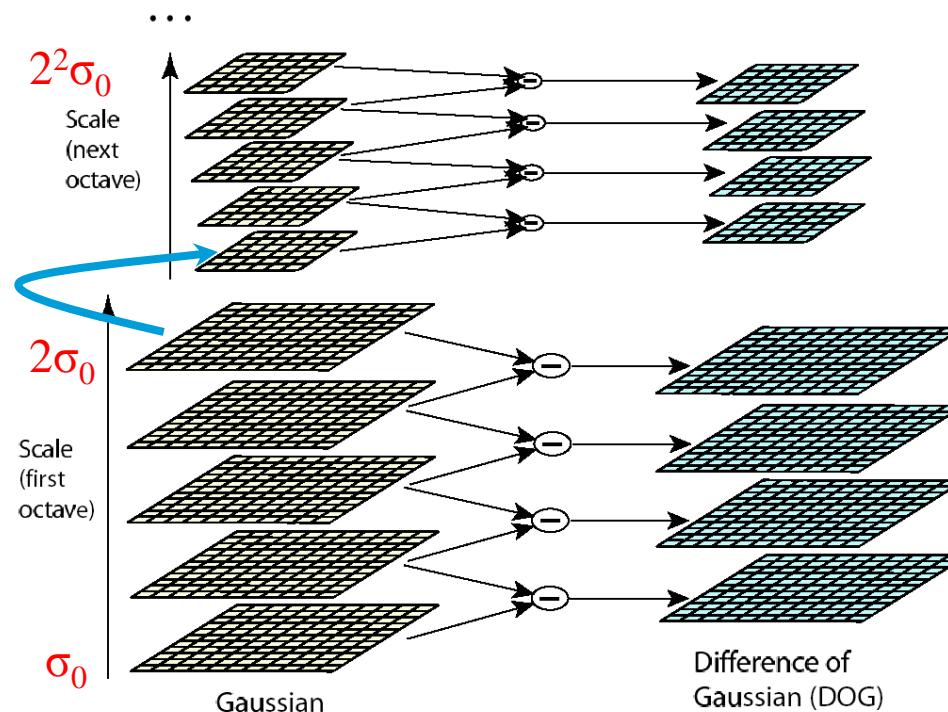
1. Scale-space Extrema Detection

(cont'd)

- DoG images are grouped by octaves
 - An octave corresponds to doubling the value of σ
- Fixed number of scales (i.e., levels) per octave



Down-sample



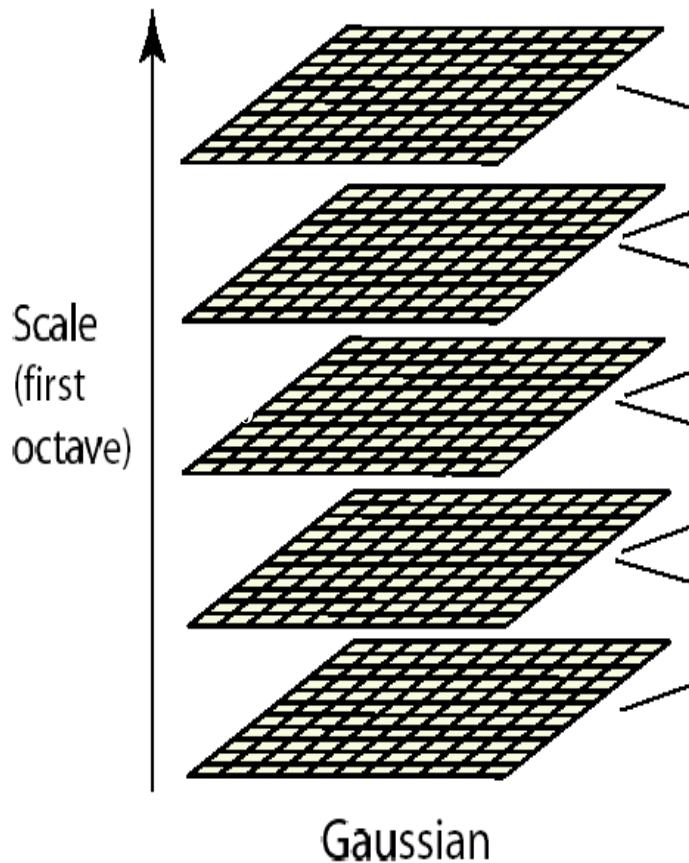
$$\begin{aligned} & G(x, y, k\sigma) - G(x, y, \sigma) \\ & \approx (k-1)\sigma^2 \nabla^2 G \end{aligned}$$

$$\begin{aligned} L(x, y, \sigma) &= \\ G(x, y, \sigma) * I(x, y) & \end{aligned}$$

$$\begin{aligned} D(x, y, \sigma) &= \\ L(x, y, k\sigma) - L(x, y, \sigma) & \end{aligned}$$

1. Scale-space Extrema Detection

(cont'd)



$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k - 1)\sigma^2 \nabla^2 G$$

- Images separated by a constant factor k
- If each octave is divided in s intervals, we have $s+1$ DoG images/octave where:

$$k^s=2 \text{ or } k=2^{1/s}$$

Note: need $(s+1)+2 = s+3$ blurred images

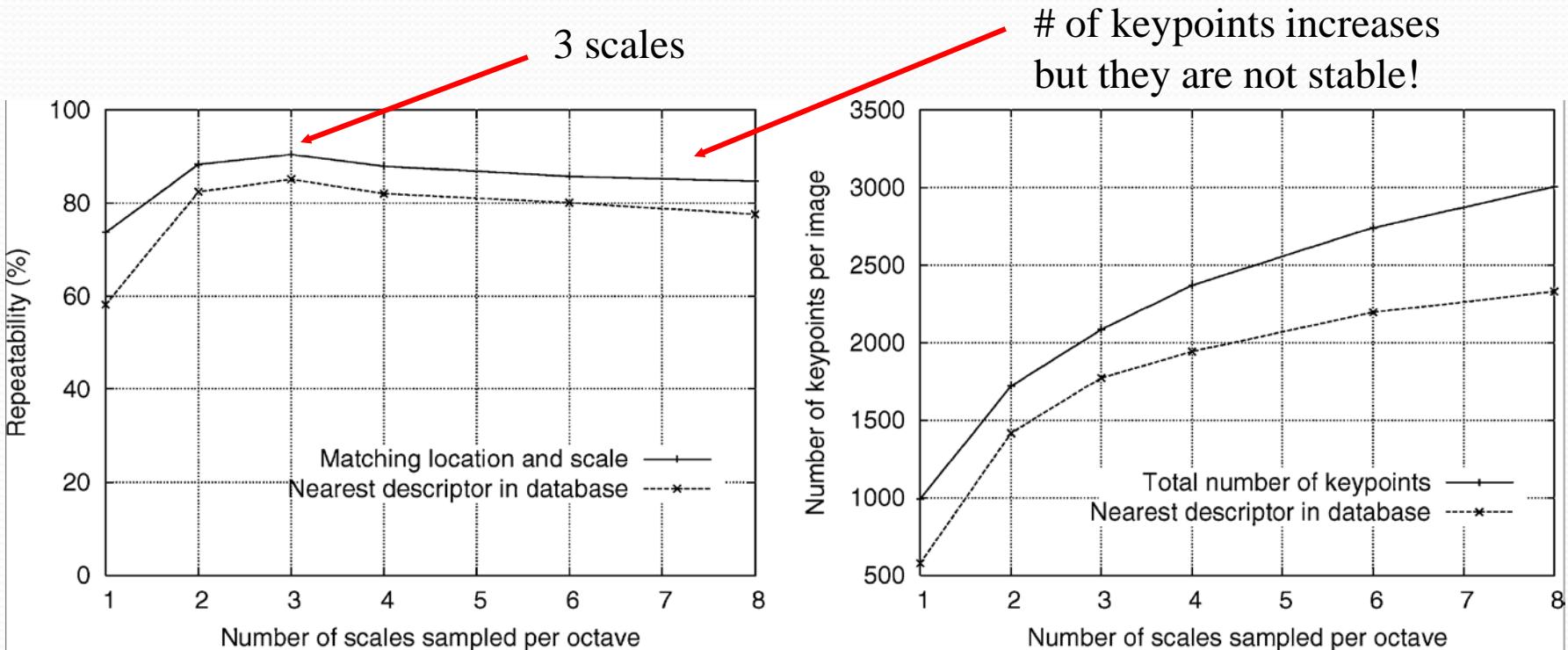
Choosing SIFT parameters

- Experimentally using a matching task:
 - 32 real images (outdoor, faces, aerial etc.)
 - Images subjected to a wide range of transformations (i.e., rotation, scaling, shear, change in brightness, noise).
 - Keypoints are detected in each image.
 - Parameters are chosen based on keypoint repeatability, localization, and matching accuracy.

1. Scale-space Extrema Detection

(cont'd)

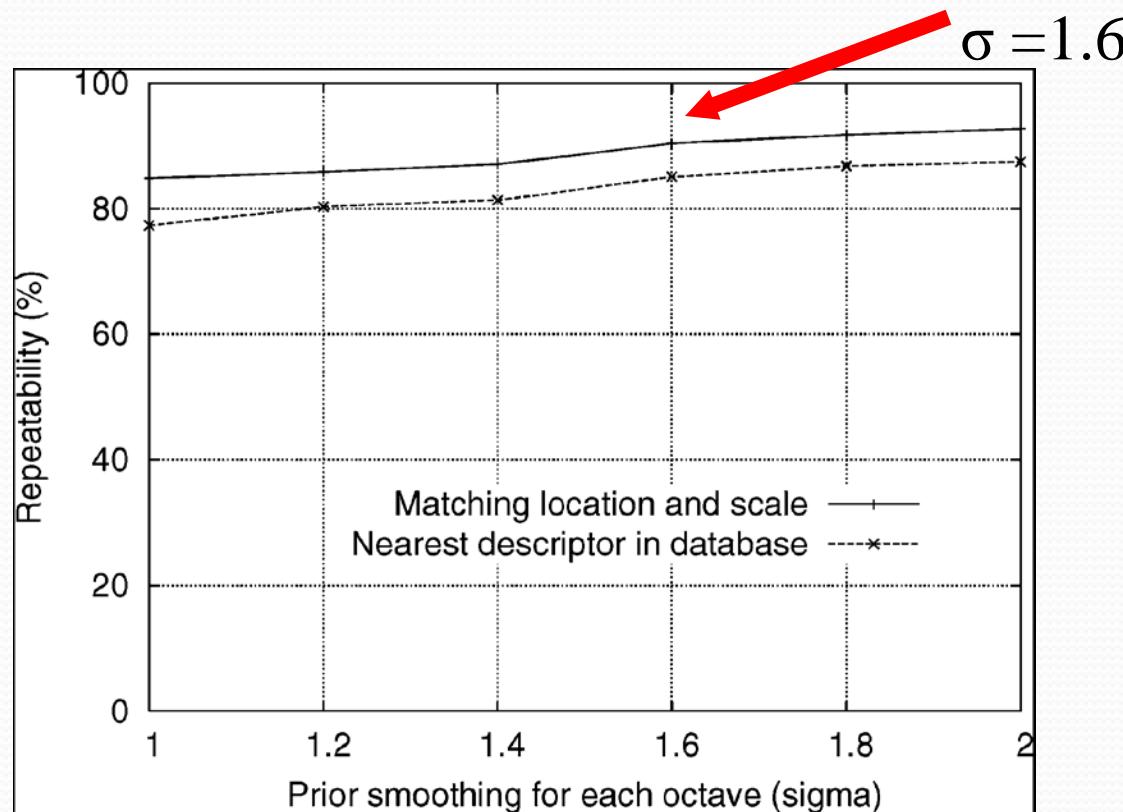
- How many scales sampled per octave?



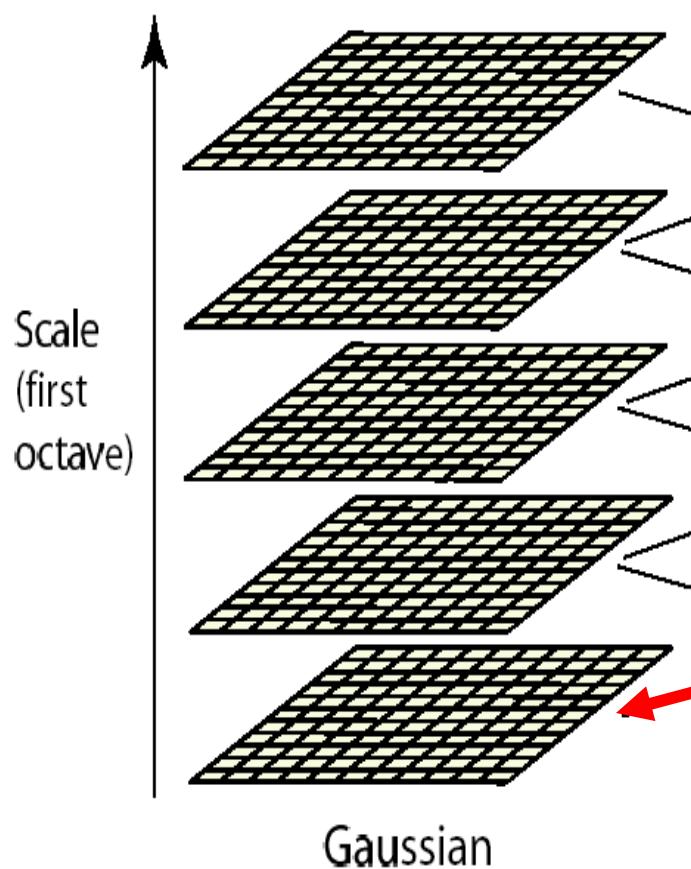
1. Scale-space Extrema Detection

(cont'd)

- Smoothing is applied to the first level of each octave.
- How to choose σ ? (i.e., integration scale)



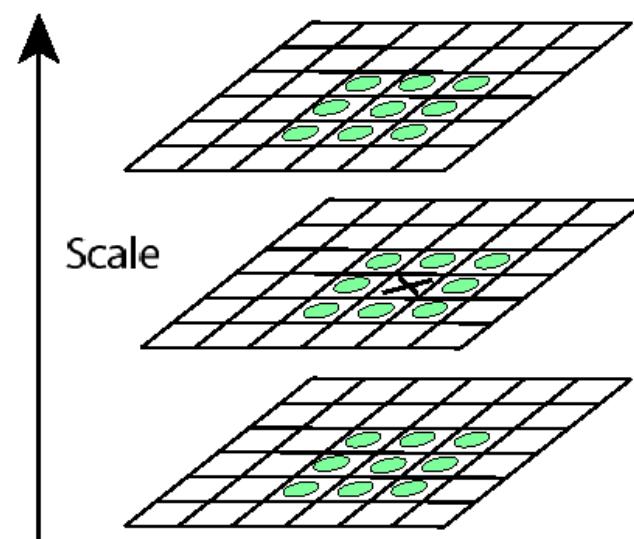
1. Scale-space Extrema Detection (cont'd)



- Pre-smoothing discards high frequencies.
- Double the size of the input image (i.e., using linear interpolation) prior to building the first level of the DoG pyramid.
- Increases the number of stable keypoints by a factor of 4.

1. Scale-space Extrema Detection (cont'd)

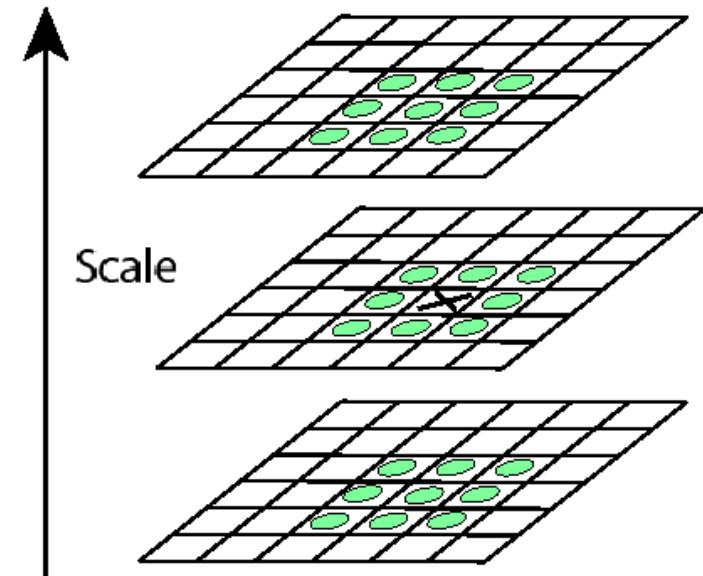
- Extract local extrema (i.e., minima or maxima) in DoG pyramid.
 - Compare each point to its 8 neighbors at the same level, 9 neighbors in the level above, and 9 neighbors in the level below (i.e., 26 total).



2. Keypoint Localization

- Determine the location and scale of keypoints to **sub-pixel** and **sub-scale** accuracy by fitting a 3D quadratic function at each keypoint.

$$X_i = (x_i, y_i, \sigma_i) \rightarrow X_i + \Delta X$$



- Substantial improvement to matching and stability!

2. Keypoint Localization

- Use Taylor expansion of $D(x,y,\sigma)$ (i.e., DoG function) around the sample point $X_i = (x_i, y_i, \sigma_i)$:

$$D(\Delta X) = D(X_i) + \frac{\partial D^T(X_i)}{\partial \mathbf{X}} \Delta \mathbf{X} + \frac{1}{2} \Delta \mathbf{X}^T \frac{\partial^2 D(X_i)}{\partial \mathbf{X}^2} \Delta \mathbf{X}$$

where

$\Delta \mathbf{X} = (x - x_i, y - y_i, \sigma - \sigma_i)$ is the offset from this point.

2. Keypoint Localization

- To find the extrema of $D(\Delta X)$: $\frac{\partial D(X_i)}{\partial X} + \frac{\partial^2 D(X_i)}{\partial X^2} \Delta X = 0$

$$\frac{\partial^2 D(X_i)}{\partial X^2} \Delta X = -\frac{\partial D(X_i)}{\partial X} \quad \rightarrow \quad \Delta X = -\frac{\partial^2 D^{-1}(X_i)}{\partial X^2} \frac{\partial D(X_i)}{\partial X}$$

- ΔX can be computed by solving a 3×3 linear system:

$$\begin{bmatrix} \frac{\partial^2 D}{\partial \sigma^2} & \frac{\partial^2 D}{\partial \sigma y} & \frac{\partial^2 D}{\partial \sigma x} \\ \frac{\partial^2 D}{\partial \sigma y} & \frac{\partial^2 D}{\partial y^2} & \frac{\partial^2 D}{\partial yx} \\ \frac{\partial^2 D}{\partial \sigma x} & \frac{\partial^2 D}{\partial yx} & \frac{\partial^2 D}{\partial x^2} \end{bmatrix} \begin{bmatrix} \Delta \sigma \\ \Delta y \\ \Delta x \end{bmatrix} = - \begin{bmatrix} \frac{\partial D}{\partial \sigma} \\ \frac{\partial D}{\partial y} \\ \frac{\partial D}{\partial x} \end{bmatrix}$$

$$\frac{\partial D}{\partial \sigma} = \frac{D_{k+1}^{i,j} - D_{k-1}^{i,j}}{2} \quad \text{use finite differences:}$$
$$\frac{\partial^2 D}{\partial \sigma^2} = \frac{D_{k-1}^{i,j} - 2D_k^{i,j} + D_{k+1}^{i,j}}{1}$$
$$\frac{\partial^2 D}{\partial \sigma y} = \frac{(D_{k+1}^{i+1,j} - D_{k-1}^{i+1,j}) - (D_{k+1}^{i-1,j} - D_{k-1}^{i-1,j})}{4}$$

2. Keypoint Localization (cont'd)

- Sub-pixel, sub-scale interpolated estimate:

$$X_i = X_i + \Delta X$$

If $\Delta X > 0.5$ in any dimension, repeat.

- Reject keypoints having low contrast.
 - i.e., sensitive to noise

If $|D(X_i + \Delta X)| < 0.03$ reject keypoint

– i.e., assumes that image values have been normalized in $[0,1]$

2. Keypoint Localization (cont'd)

- Reject points lying on edges (or being close to edges)
- Harris uses the 2nd order moment matrix:

$$A_W(x, y) = \sum_{x \in W, y \in W} \begin{bmatrix} f_x^2 & f_x f_y \\ f_x f_y & f_y^2 \end{bmatrix}$$

$$\mathbf{R}(A_W) = \det(A_W) - \alpha \operatorname{trace}^2(A_W)$$

or $\mathbf{R}(A_W) = \lambda_1 \lambda_2 - \alpha (\lambda_1 + \lambda_2)^2$

2. Keypoint Localization (cont'd)

- SIFT uses the Hessian matrix for efficiency.
 - i.e., encodes principal curvatures

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \quad \begin{aligned} \alpha &: \text{largest eigenvalue } (\lambda_{\max}) \\ \beta &: \text{smallest eigenvalue } (\lambda_{\min}) \\ &\text{(proportional to principal curvatures)} \end{aligned}$$

$$\begin{aligned} \text{Tr}(\mathbf{H}) &= D_{xx} + D_{yy} = \alpha + \beta, \\ \text{Det}(\mathbf{H}) &= D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta. \end{aligned} \rightarrow \frac{\text{Tr}(\mathbf{H})^2}{\text{Det}(\mathbf{H})} = \frac{(\alpha + \beta)^2}{\alpha\beta} = \frac{(r\beta + \beta)^2}{r\beta^2} = \frac{(r+1)^2}{r},$$

(r = α/β)

$$\frac{\text{Tr}(\mathbf{H})^2}{\text{Det}(\mathbf{H})} < \frac{(r+1)^2}{r} \quad (\text{SIFT uses } r = 10)$$

2. Keypoint Localization (cont'd)



(a) 233×189 image

(b) 832 DoG extrema

(c) 729 left after low contrast threshold

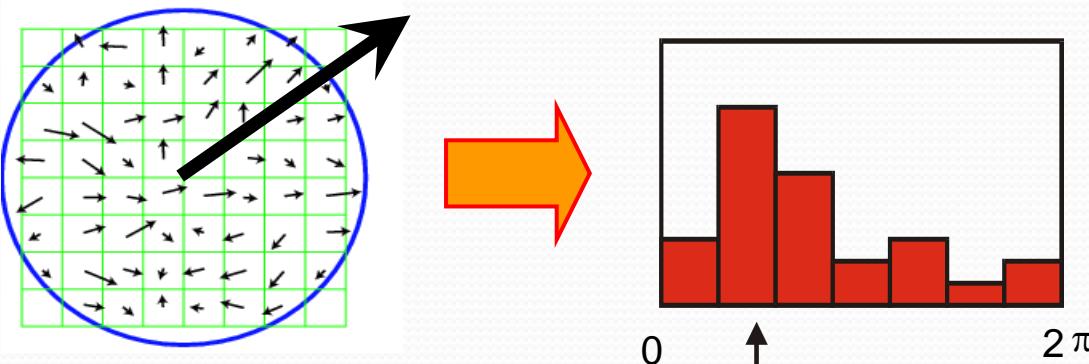
(d) 536 left after testing ratio based on Hessian

3. Orientation Assignment

- Create histogram of gradient directions, within a region around the keypoint, at selected scale (i.e., scale invariance): $L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

$$\theta(x, y) = a \tan 2((L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y)))$$

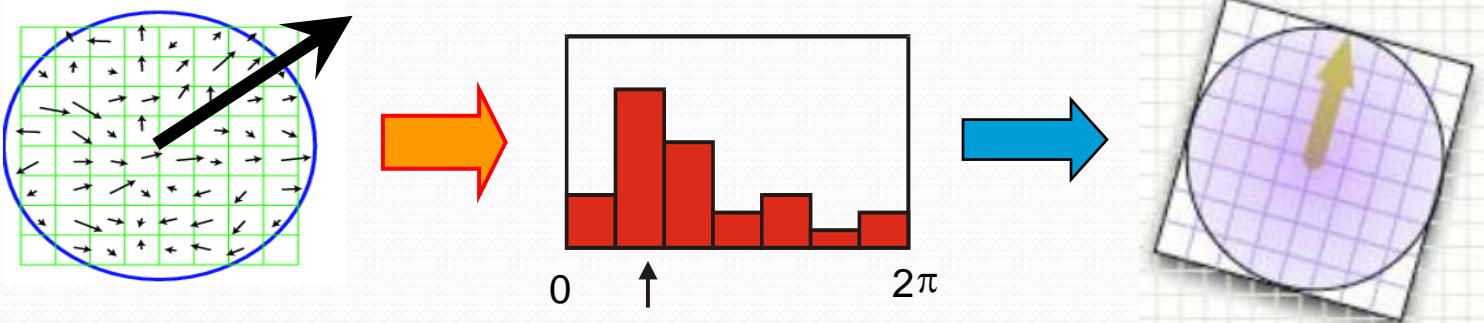


36 bins (i.e., 10° per bin)

- Histogram entries **are weighted** by (i) gradient magnitude and (ii) a Gaussian function with σ equal to 1.5 times the scale of the keypoint.

3. Orientation Assignment (cont'd)

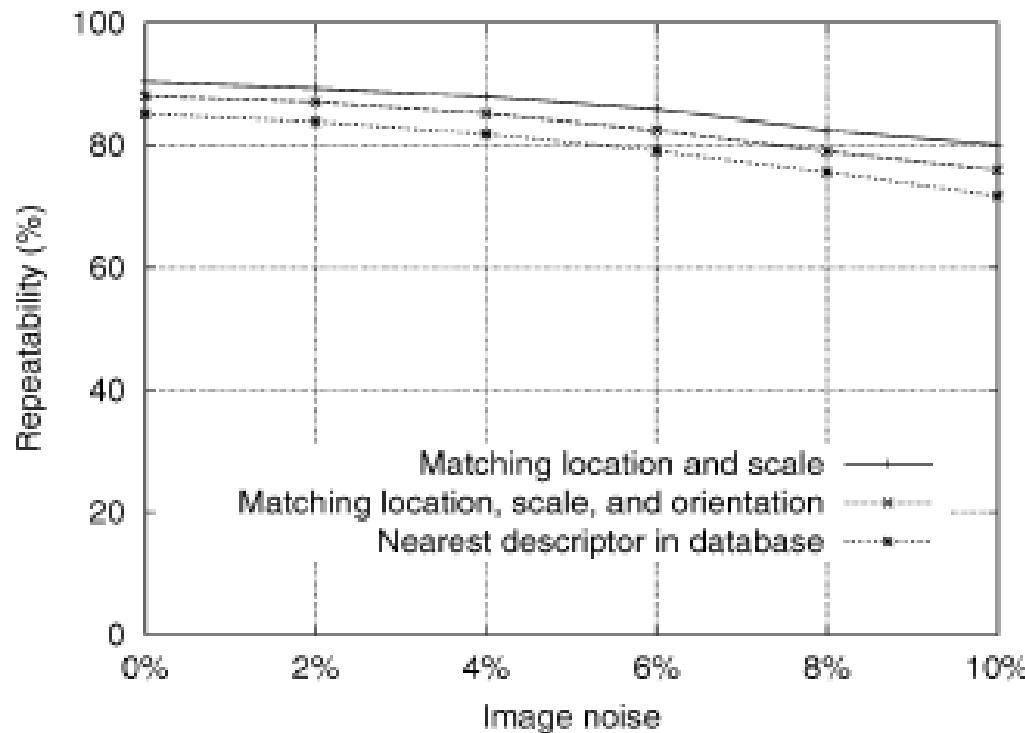
- Assign canonical orientation at **peak** of smoothed histogram (fit parabola to better localize peak).



- In case of peaks within 80% of highest peak, multiple orientations assigned to keypoints.
 - About 15% of keypoints have multiple orientations assigned.
 - Significantly improves stability of matching.

3. Orientation Assignment (cont'd)

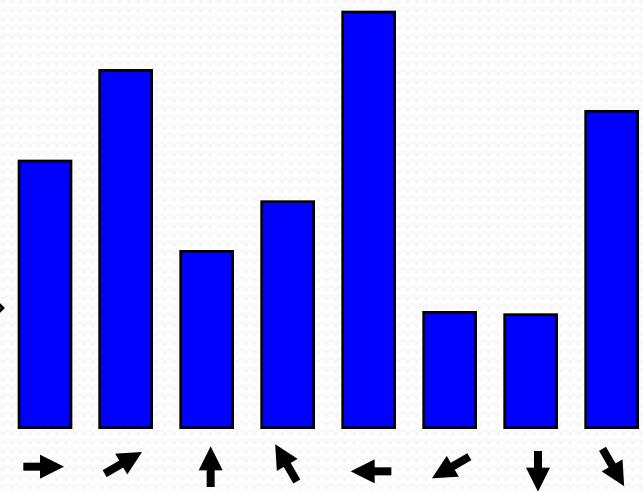
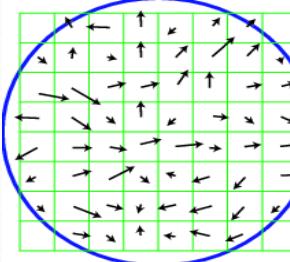
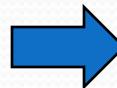
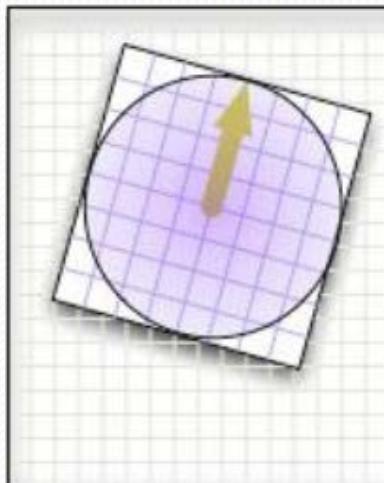
- Stability of location, scale, and orientation (within 15 degrees) under noise.



4. Keypoint Descriptor

- Have achieved invariance to location, scale, and orientation.
- Next, tolerate illumination and viewpoint changes.

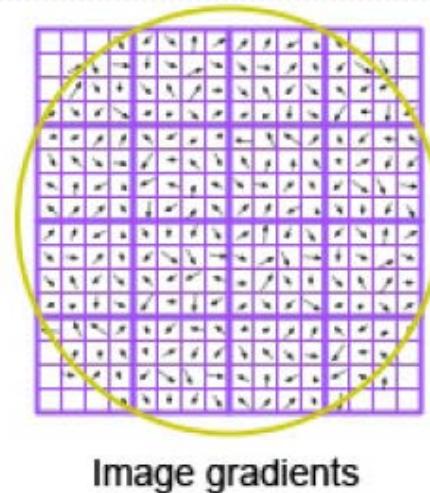
Orientation histogram of gradient magnitudes



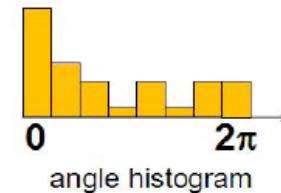
8 bins

4. Keypoint Descriptor (cont'd)

1. Take a 16 x 16 window around detected interest point.

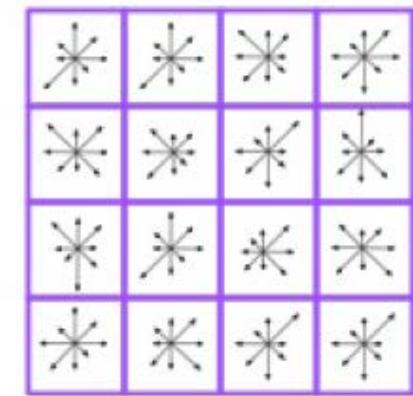
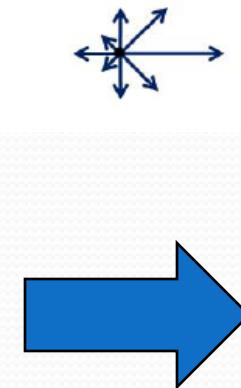


2. Divide into a 4x4 grid of cells.



(8 bins)

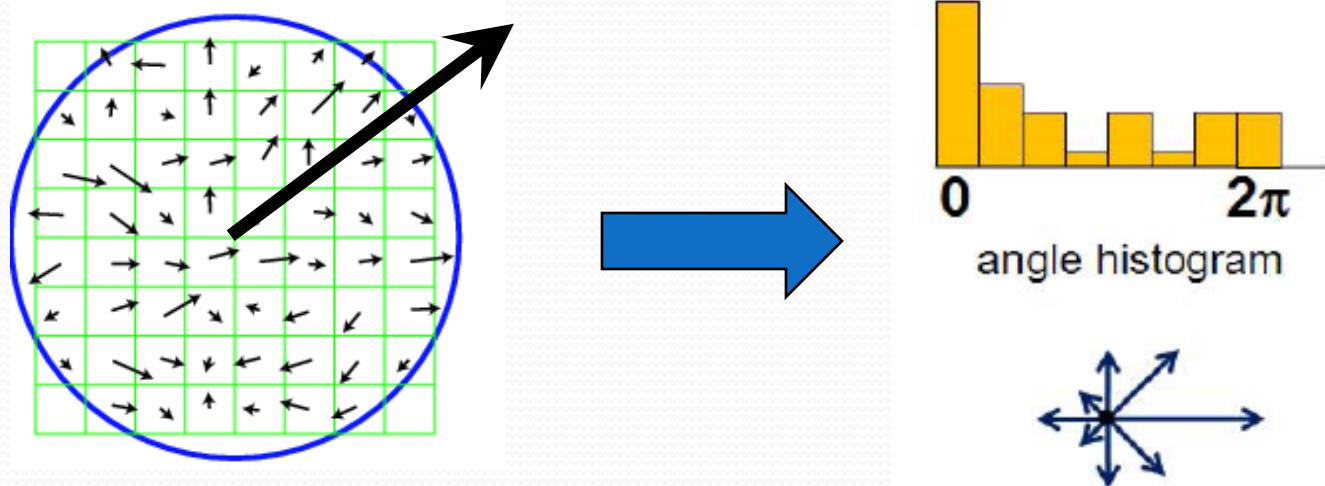
3. Compute histogram in each cell.



16 histograms x 8 orientations
= 128 features

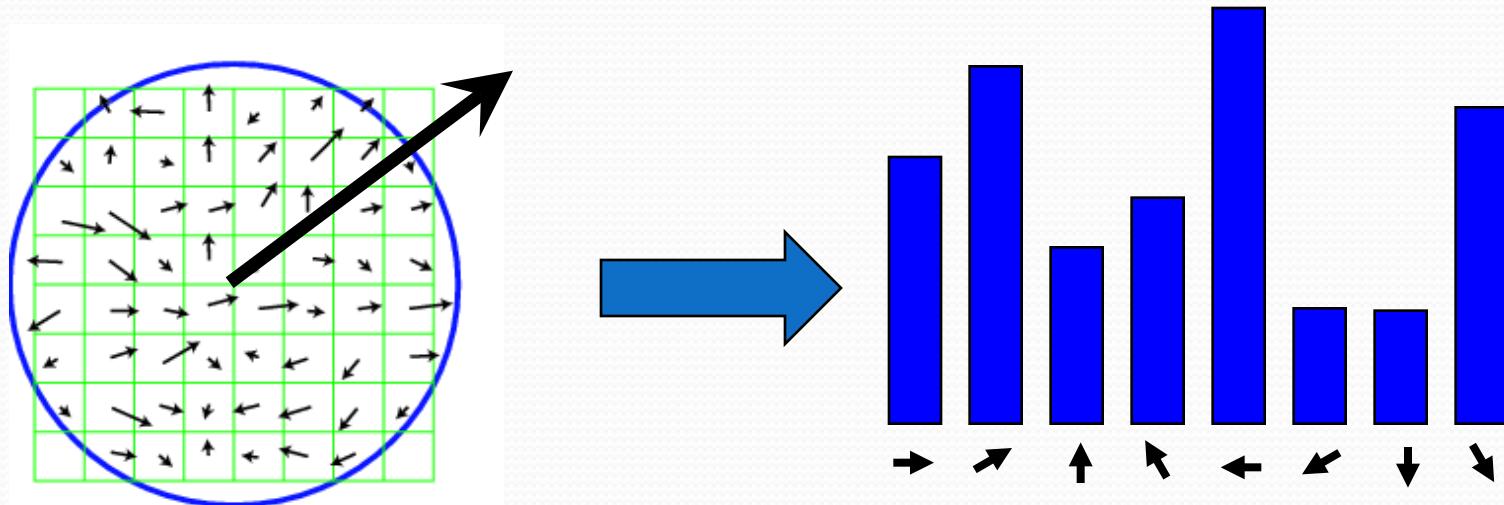
4. Keypoint Descriptor (cont'd)

- Each histogram entry **is weighted** by (i) gradient magnitude and (ii) a Gaussian function with σ equal to 0.5 times the width of the descriptor window.



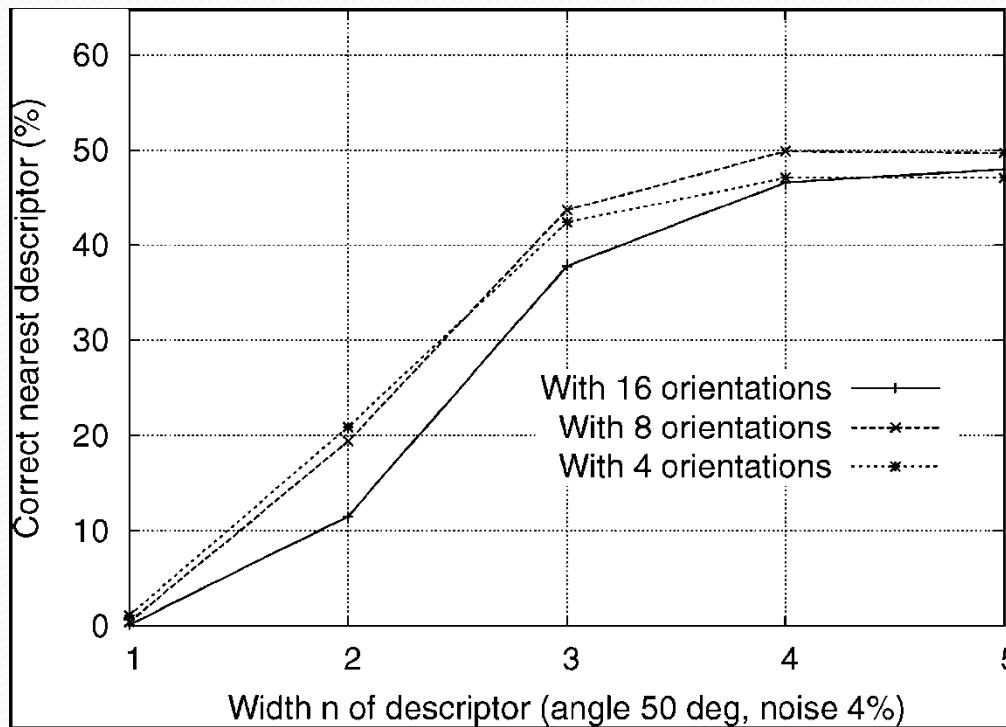
4. Keypoint Descriptor (cont'd)

- **Partial Voting:** distribute histogram entries into adjacent bins (i.e., **additional robustness to shifts**)
 - Each entry is added to all bins, multiplied by a weight of $1-d$, where d is the distance from the bin it belongs.

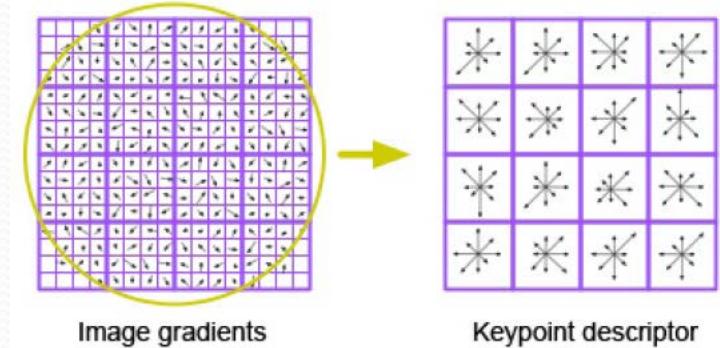


4. Keypoint Descriptor (cont'd)

- Descriptor depends on two parameters:
 - (1) number of orientations r
 - (2) $n \times n$ array of orientation histograms
- $\left. \right\} r n^2 \text{ features}$

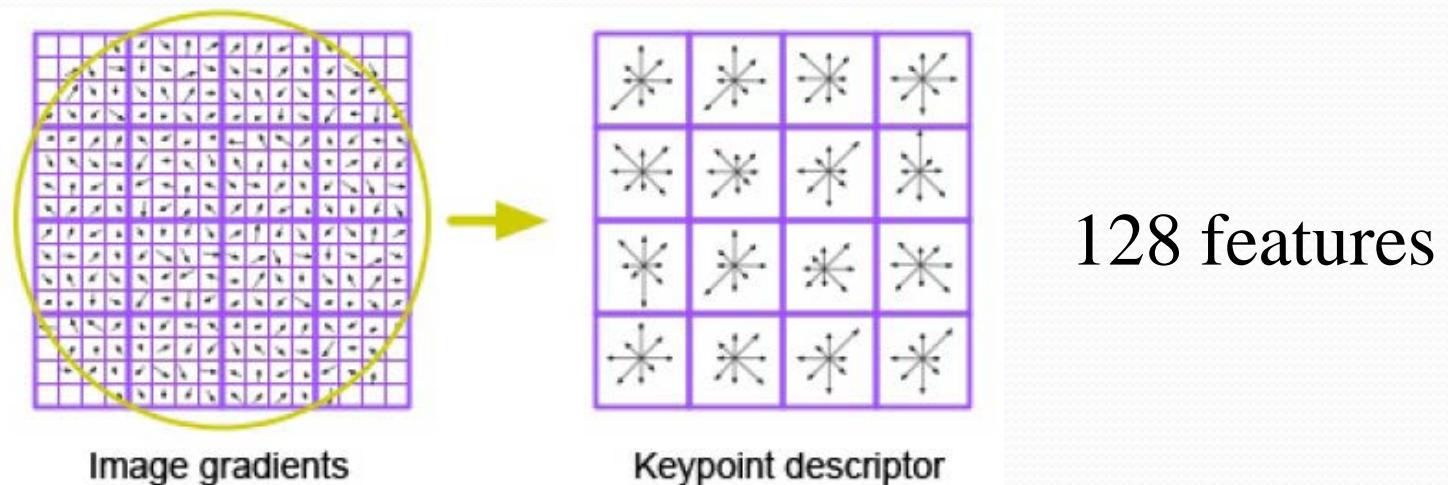


SIFT: $r = 8$, $n = 4$
128 features



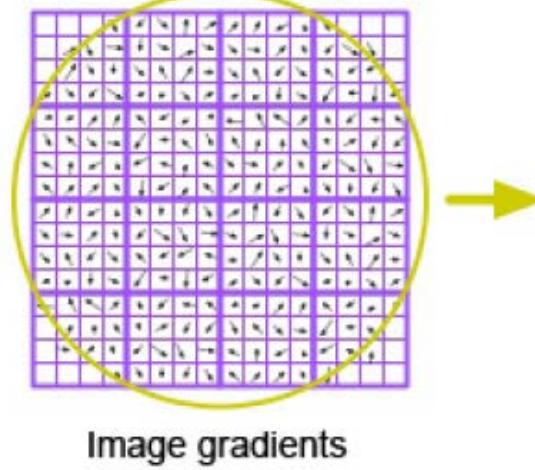
4. Keypoint Descriptor (cont'd)

- Invariance to affine (linear) illumination changes:
 - Normalization to **unit length** is sufficient.



4. Keypoint Descriptor (cont'd)

- Non-linear illumination changes :
 - Saturation affects gradient magnitudes more than orientations
 - Threshold gradient magnitudes to be no larger than 0.2 and renormalize to **unit length**
(i.e., emphasizes gradient orientations than magnitudes)



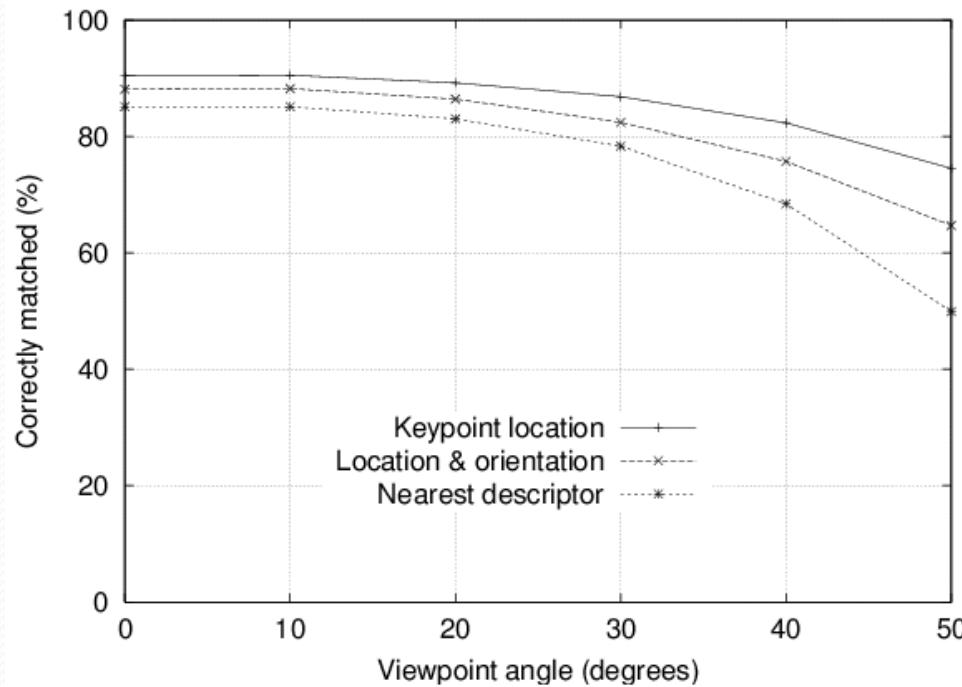
*	*	*	*
*	*	*	*
*	*	*	*
*	*	*	*

Keypoint descriptor

128 features

Robustness to viewpoint changes

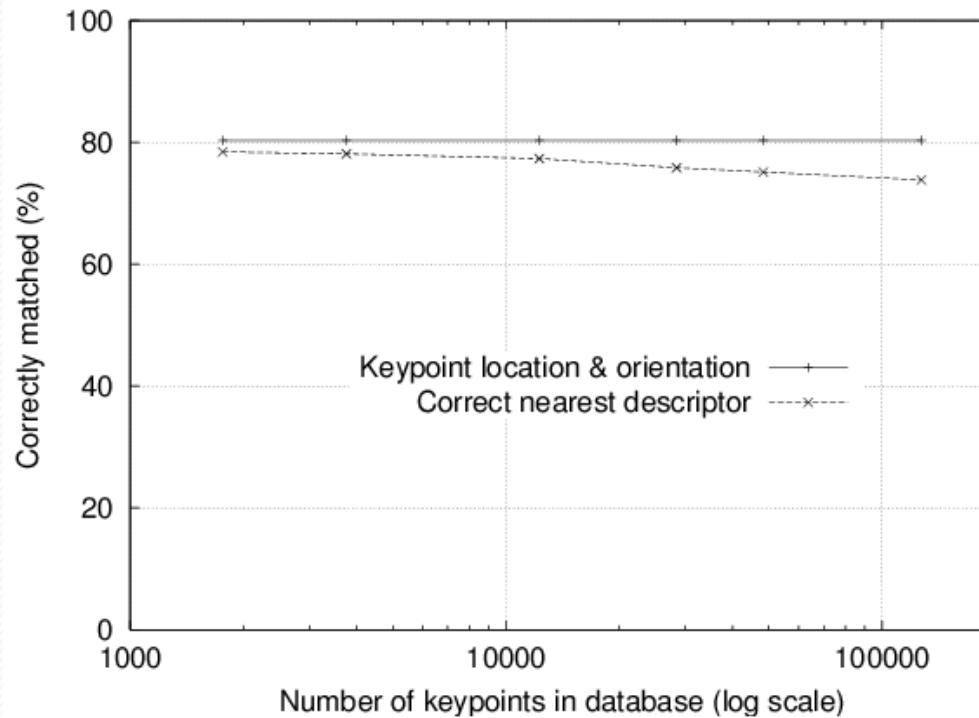
- Match features after random change in image scale and orientation, with 2% image noise, and affine distortion.
- Find nearest neighbor in database of 30,000 features.



Additional robustness can be achieved using affine invariant region detectors.

Distinctiveness

- Vary size of database of features, with 30 degree affine change, 2% image noise.
- Measure % correct for single nearest neighbor match.

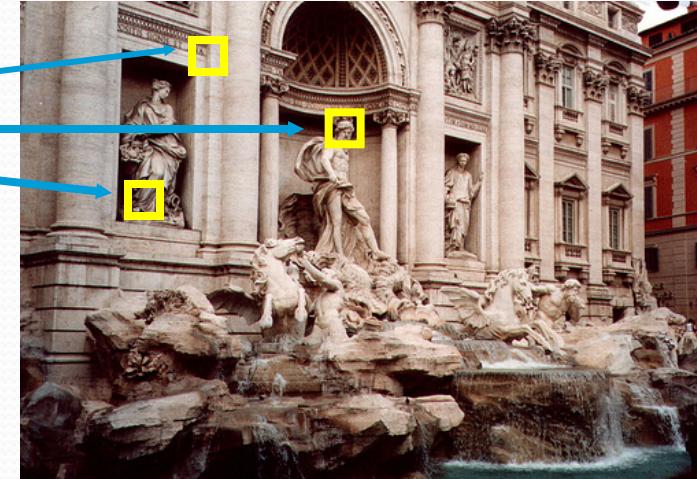


Matching SIFT features

- Given a feature in I_1 , how to find the best match in I_2 ?
 - Define distance function that compares two descriptors.
 - Test all the features in I_2 , find the one with min distance.



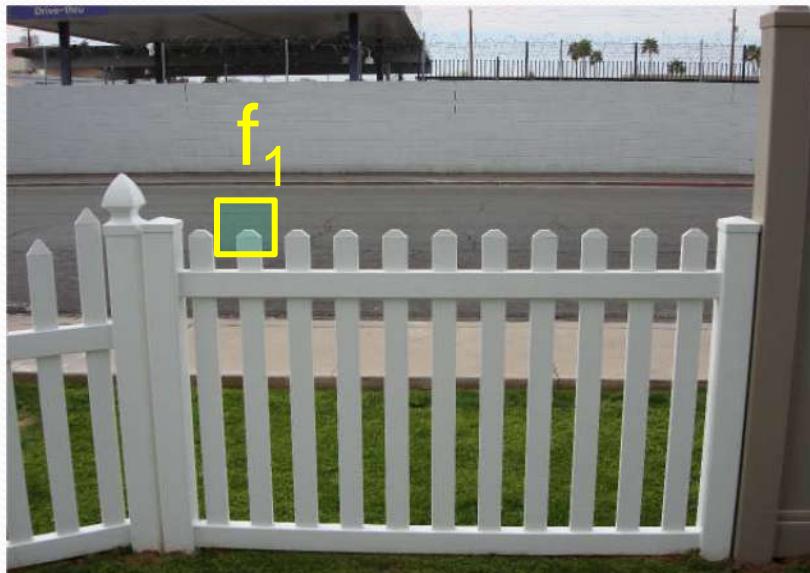
I_1



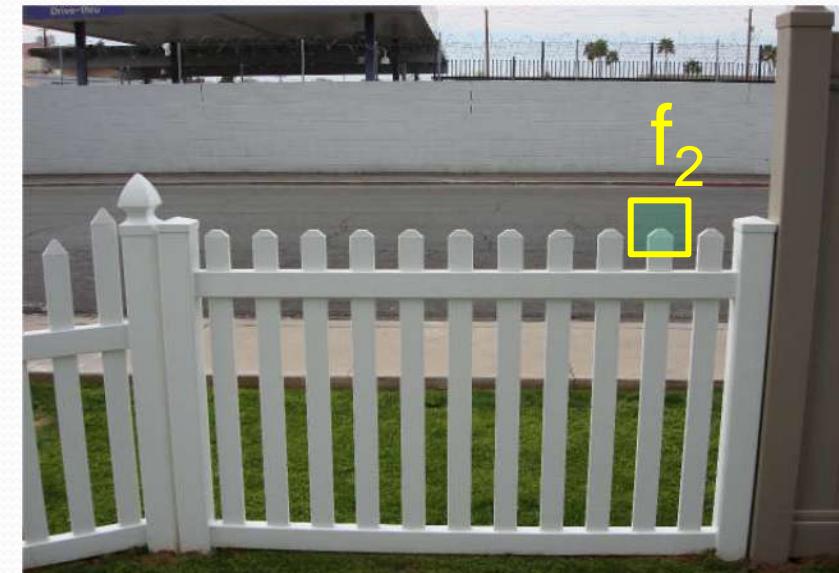
I_2

Matching SIFT features (cont'd)

- How to define the distance between two features f_1 , f_2 ?
 - Simple approach: SSD(f_1 , f_2) (i.e., sum of squared differences)
 - Can give good scores to very ambiguous (bad) matches ?



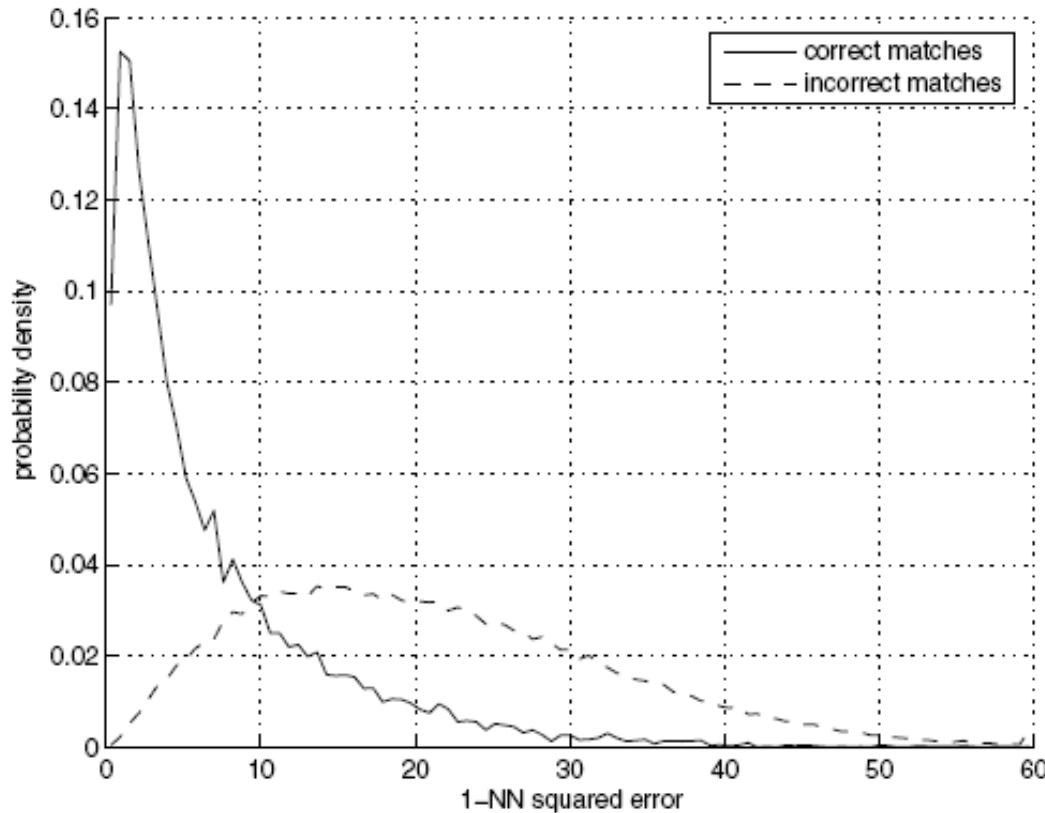
I_1



I_2

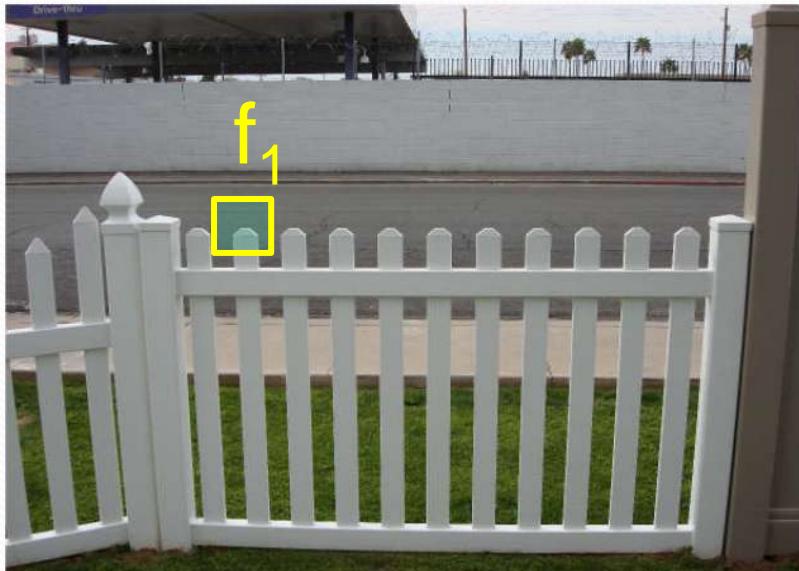
Matching SIFT features (cont'd)

$\text{SSD}(f_1, f_2) < t$ How to set t ?

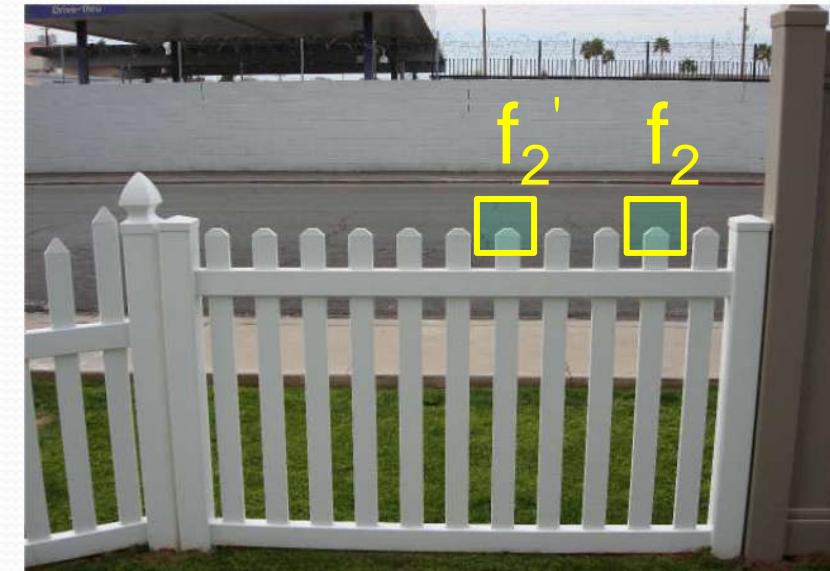


Matching SIFT features (cont'd)

- How to define the difference between two features f_1 , f_2 ?
 - Better approach: $\text{SSD}(f_1, f_2) / \text{SSD}(f_1, f'_2)$
 - f_2 is best SSD match to f_1 in I_2
 - f'_2 is 2nd best SSD match to f_1 in I_2



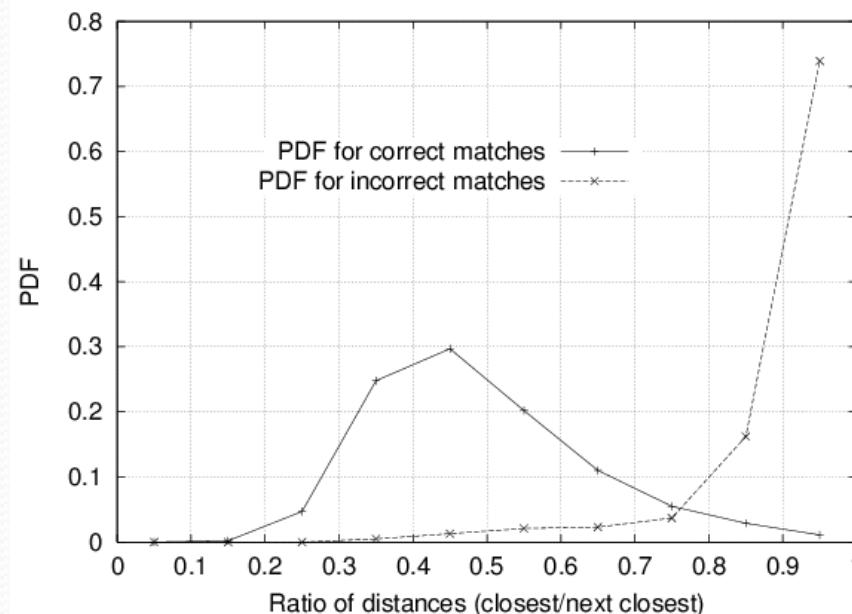
I_1



I_2

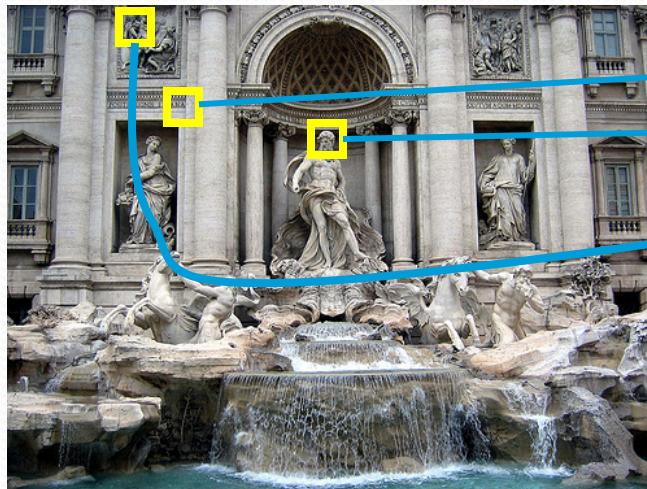
Matching SIFT features (cont'd)

- Accept a match if $\text{SSD}(f_1, f_2) / \text{SSD}(f_1, f'_2) < t$
- $t=0.8$ has given good results in object recognition.
 - Eliminated 90% of false matches.
 - Discarded less than 5% of correct matches

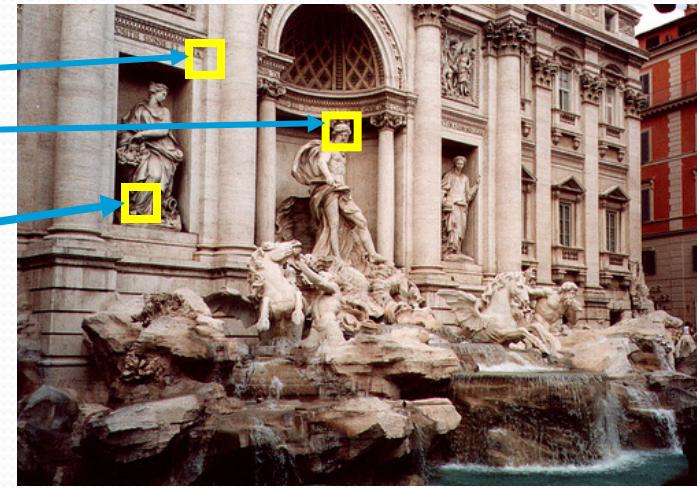


Matching SIFT features (cont'd)

- How to evaluate the performance of a feature matcher?

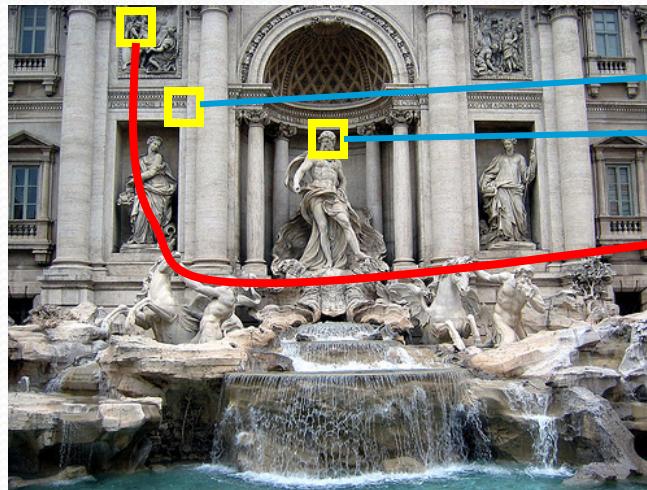


50
75
200

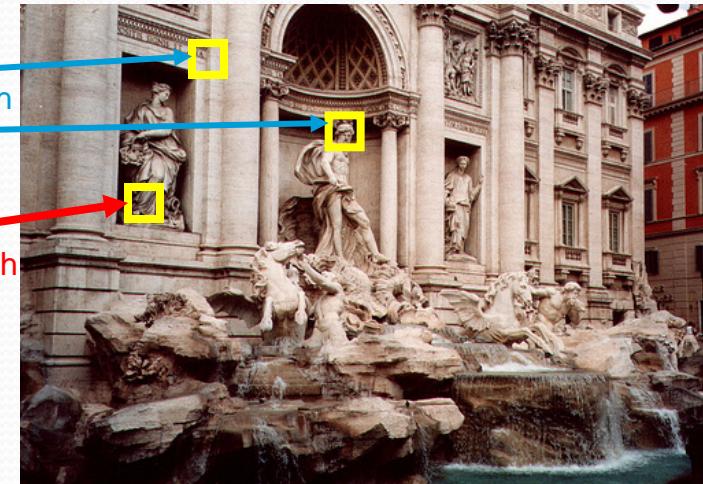


Matching SIFT features (cont'd)

- Threshold t affects # of correct/false matches



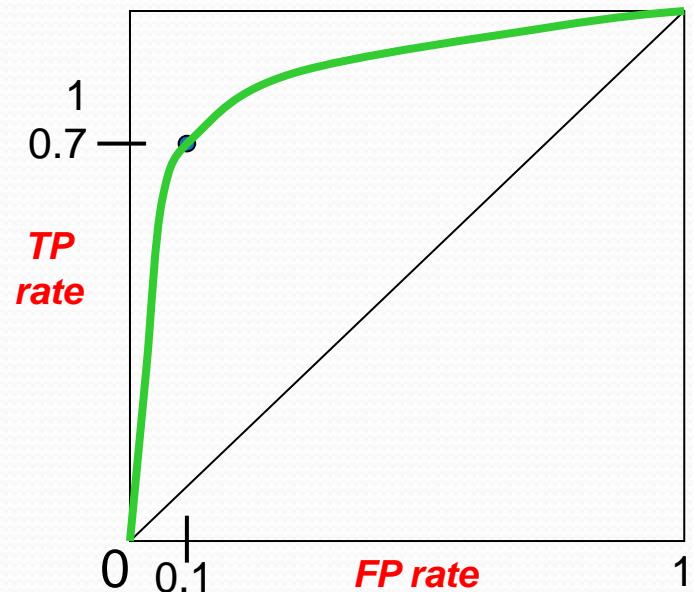
50
true match
75
200
false match



- True positives (TP)** = # of detected matches that are correct
- False positives (FP)** = # of detected matches that are incorrect

Matching SIFT features(cont'd)

- ROC Curve
 - Generated by computing (FP, TP) for different thresholds.
 - Need to maximize area under the curve (AUC)



http://en.wikipedia.org/wiki/Receiver_operating_characteristic

SURF: Speeded Up Robust Features

- Fast implementation of a variation of the SIFT descriptor.
- Key idea: fast approximation of (i) Hessian matrix and (ii) descriptor using “integral images”.

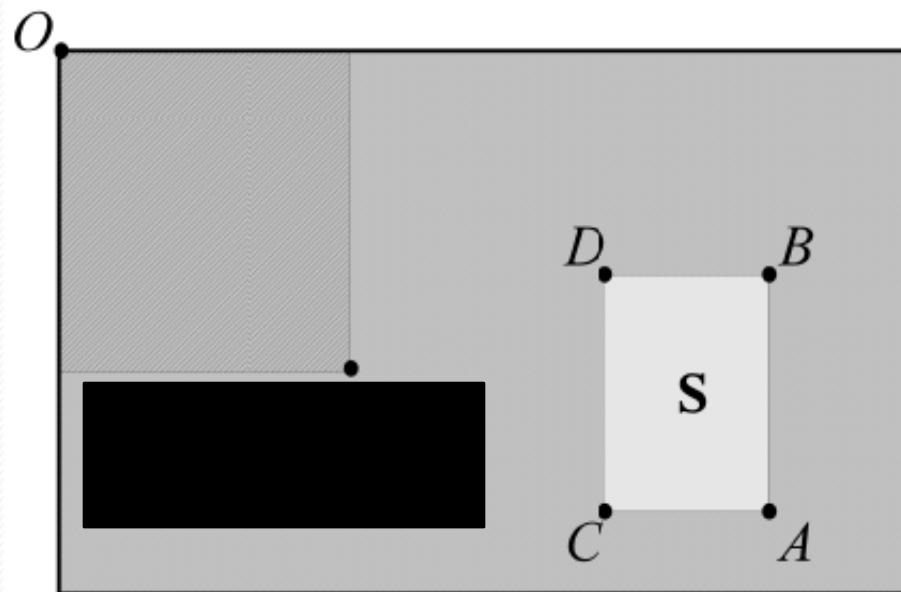
$$\mathcal{H}(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix},$$

- What is an “integral image”?

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool, “SURF: Speeded Up Robust Features”, European Computer Vision Conference (ECCV), 2006.

Integral Image

- The integral image $I_{\Sigma}(x,y)$ of an image $I(x, y)$ represents the sum of all pixels in $I(x, y)$ of a rectangular region formed by $(0,0)$ and (x,y) .



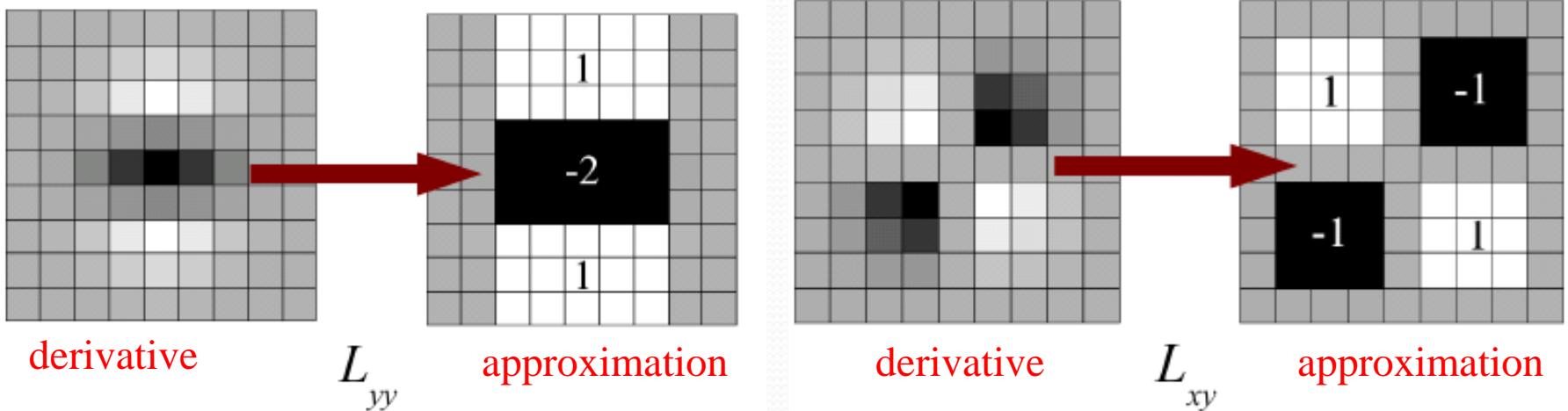
Using integral images, it takes only **four** array references to calculate the sum of pixels over a rectangular region of any size.

$$S = A - B - C + D$$

SURF: Speeded Up Robust Features (cont'd)

- Approximate L_{xx} , L_{yy} , and L_{xy} using box filters.

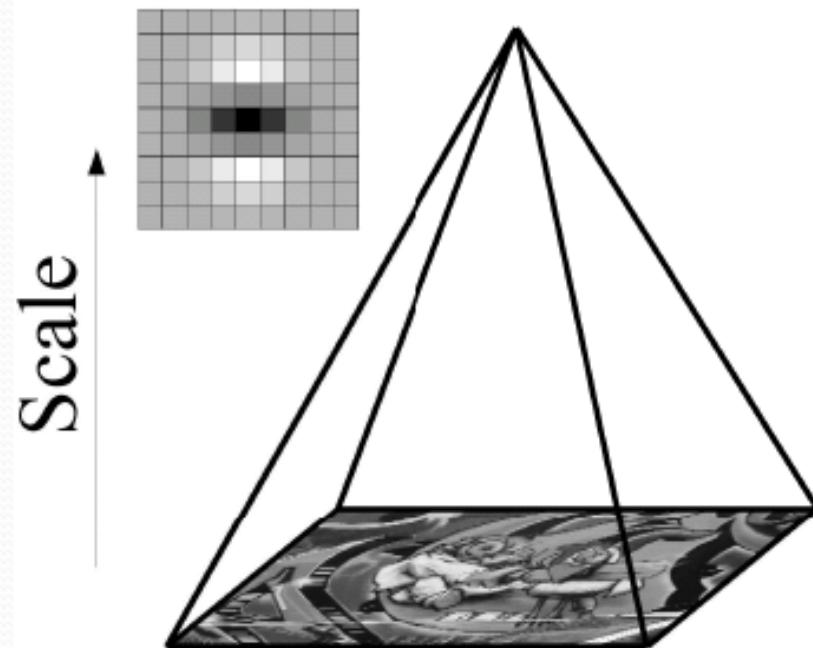
(box filters shown are 9×9 – good approximations for a Gaussian with $\sigma=1.2$)



- Can be computed very fast using integral images!

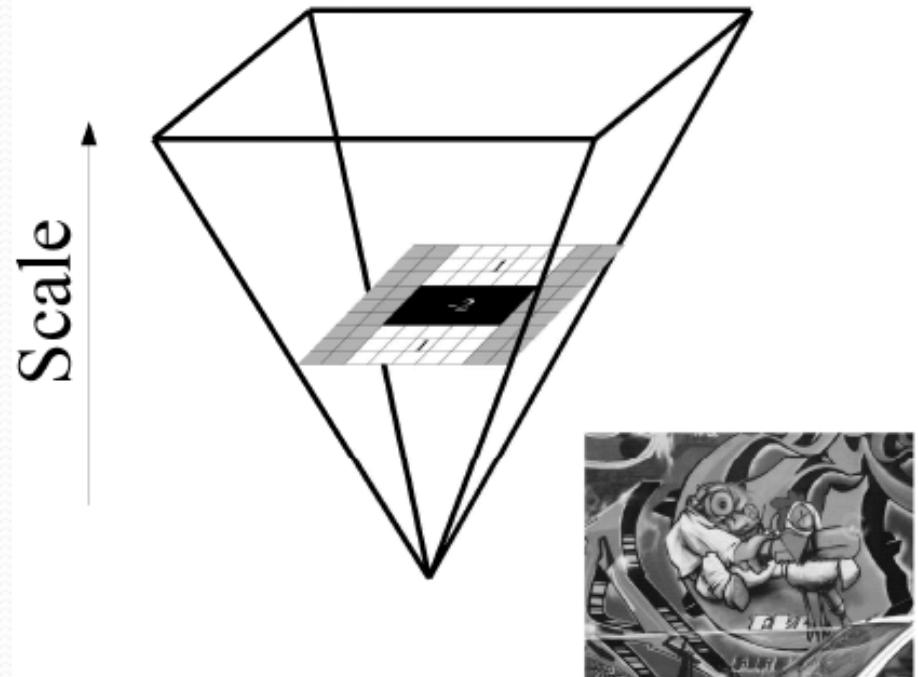
SURF: Speeded Up Robust Features (cont'd)

- In SIFT, images are repeatedly smoothed with a Gaussian and subsequently subsampled in order to achieve a higher level of the pyramid.



SURF: Speeded Up Robust Features (cont'd)

- Alternatively, we can use filters of larger size on the original image.
- Due to using integral images, filters of any size can be applied at exactly the same speed!



(see paper for details on how the filter size and octaves are computed)

SURF: Speeded Up Robust Features (cont'd)

- Approximation of H:

Using DoG

$$\mathcal{H}(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix},$$

Using box filters

$SIFT : H_{approx}^{SIFT} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix}$

$SURF : H_{approx}^{SURF} = \begin{bmatrix} \hat{L}_{xx} & \hat{L}_{xy} \\ \hat{L}_{yx} & \hat{L}_{yy} \end{bmatrix}$



SURF: Speeded Up Robust Features (cont'd)

- Instead of using a different measure for selecting the location and scale of interest points (like in SIFT(), SURF uses the determinant of H_{approx}^{SURF} to find both.
- Determinant elements must be weighted to obtain a good approximation of $\det(H)$:

$$\det(H_{approx}^{SURF}) = \hat{L}_{xx}\hat{L}_{yy} - (0.9\hat{L}_{xy})^2$$

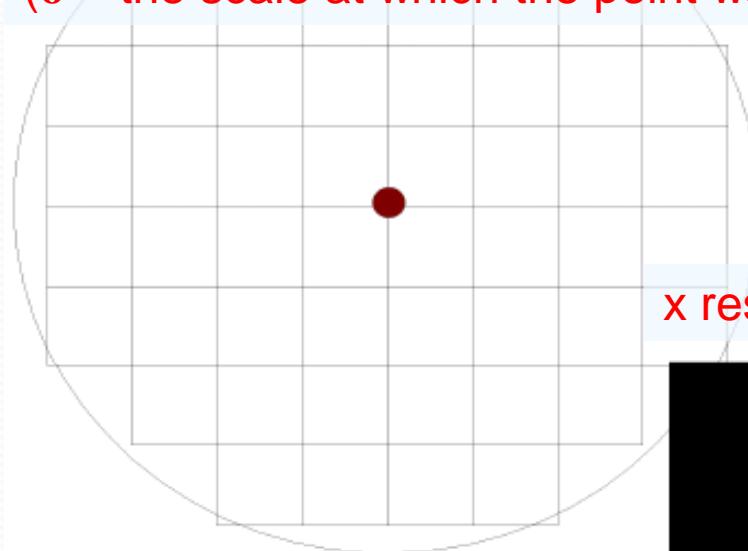
SURF: Speeded Up Robust Features (cont'd)

- Once interest points have been localized both in space and scale, the next steps are:
 - (1) Orientation assignment
 - (2) Keypoint descriptor

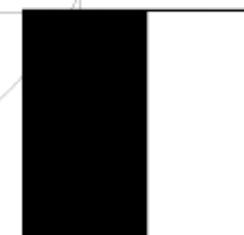
SURF: Speeded Up Robust Features (cont'd)

- Orientation assignment

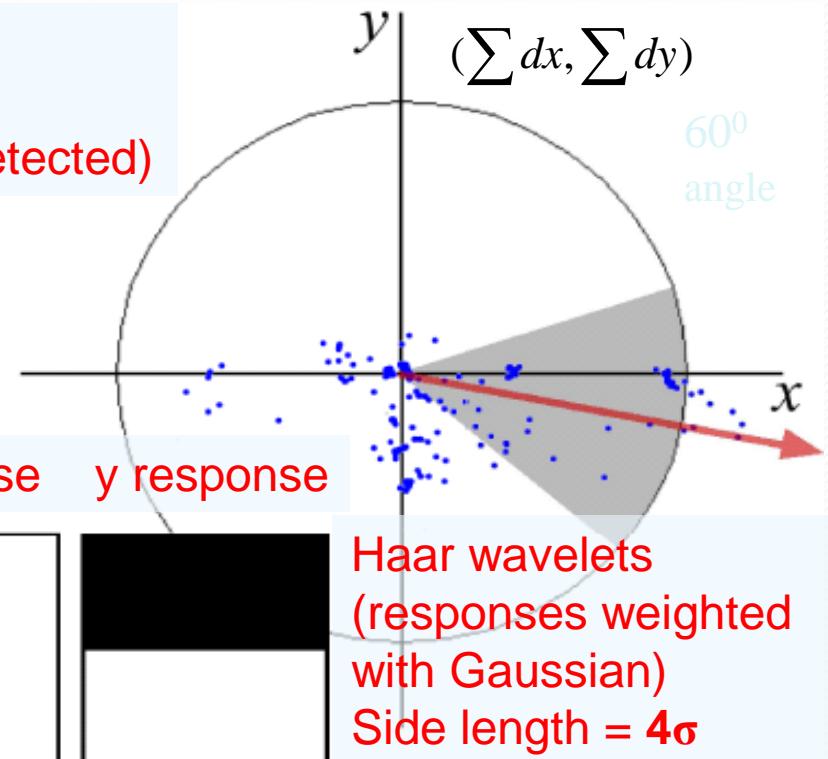
Circular neighborhood of radius 6σ around the interest point (σ = the scale at which the point was detected)



x response



y response

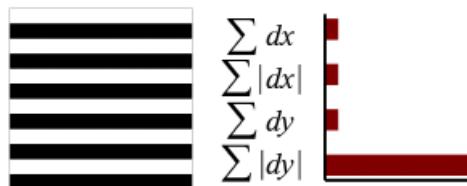
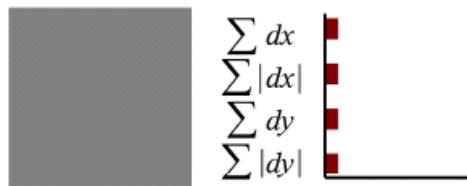


Can be computed very fast using integral images!

SURF: Speeded Up Robust Features (cont'd)

- Keypoint descriptor (square region of size 20σ)

- Description

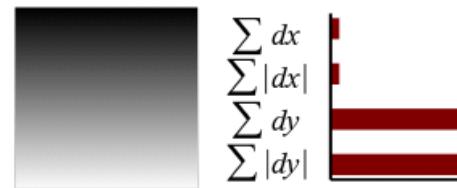
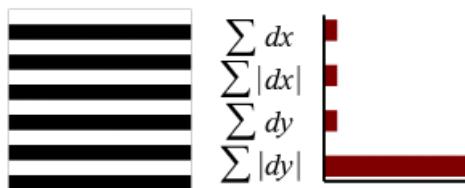
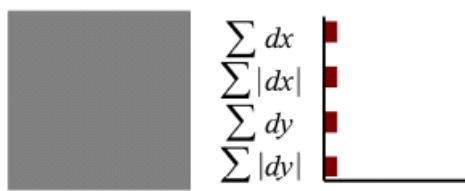
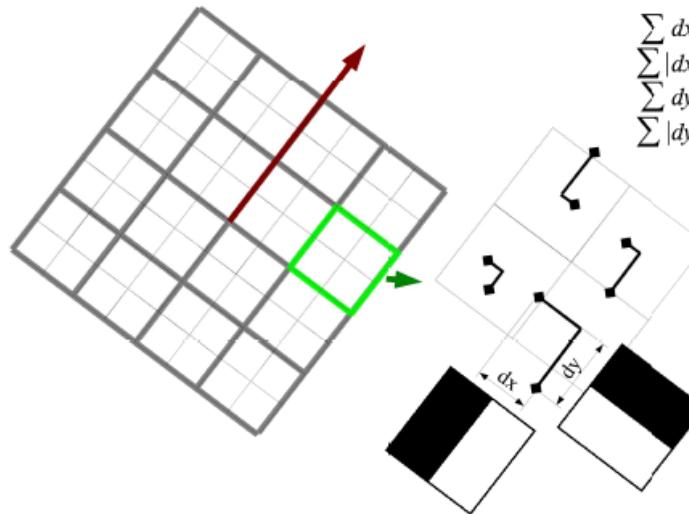


- Sum the response over each sub-region for d_x and d_y separately.
- To bring in information about the polarity of the intensity changes, extract the sum of absolute value of the responses too.

**Feature vector size:
4 x 16 = 64**

SURF: Speeded Up Robust Features (cont'd)

- Description



- SURF-128
 - The sum of d_x and $|d_x|$ are computed separately for points where $d_y < 0$ and $d_y > 0$
 - Similarly for the sum of d_y and $|d_y|$
 - **More discriminatory!**

SURF: Speeded Up Robust Features

- Has been reported to be 3 times faster than SIFT.
- Less robust to illumination and viewpoint changes compared to SIFT.

K. Mikolajczyk and C. Schmid, "A Performance Evaluation of Local Descriptors", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 10, pp. 1615-1630, 2005.