

Chapitre 7 : Extraction de primitives Partie 2 – Détection de caractéristiques

Joël Lefebvre (UQÀM)

INF600F – Traitement d'images

Automne 2024

Annonces

- Évaluation des enseignements
 - Date limite : 2 décembre
 - <https://portailetudiant.uqam.ca>
- **TP3** : remise ce dimanche (24 novembre) avant 23h59
- **TP4** : remise avant le dimanche 15 décembre avant 23h59
- **Examen final**
 - Mercredi, 11 décembre entre 14h à 17h
 - 1 feuille de note manuscrite de format Lettre (8 ½ x 11)
 - **Plus de détails à venir**
- Cours de l'hiver
 - **INF5071** : Infographie
 - **INF889G** : Vision par ordinateur

Sommaire

- Retour sur la détection de coins
- Amélioration de la détection de points caractéristiques
- Description de points caractéristiques
- Pairage

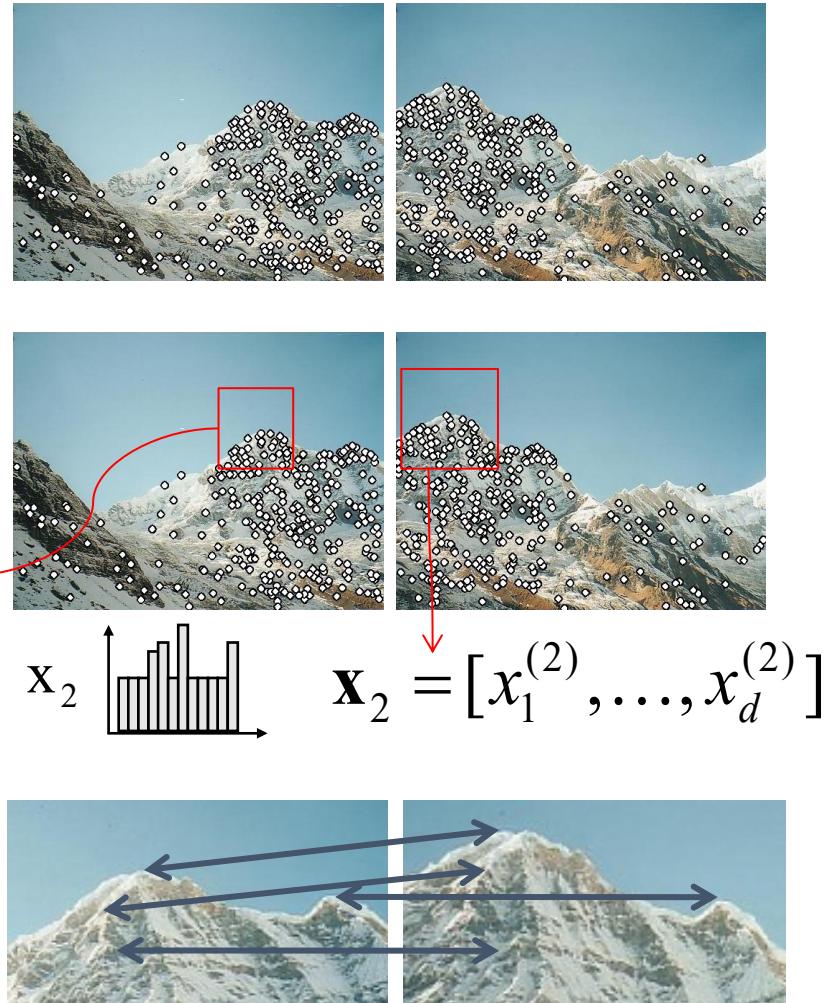
Références

- Chityala2020 : Chapitre 10 : *Image Measurements*
- Gonzalez, Ch12, Section 12.6 : *Whole-Image Features*
- Szeliski, Ch4, Section 4.1 : *Points and Patches*
- Burger, Vol.3, Ch7 : *SIFT – Scale-Invariant Local Features*

Rappel : Approche

- **Détection** : Identifier les points d'intérêt
- **Description** : Extraire un vecteur de caractéristiques (descripteur) autour de chaque point d'intérêt $\mathbf{x}_1 = [x_1^{(1)}, \dots, x_d^{(1)}]$
- **Pairage** : Déterminer la correspondance entre les descripteurs des deux points de vue.

$$d(x_1, x_2) < T$$

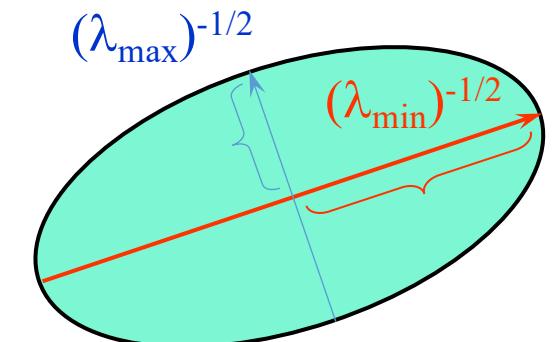
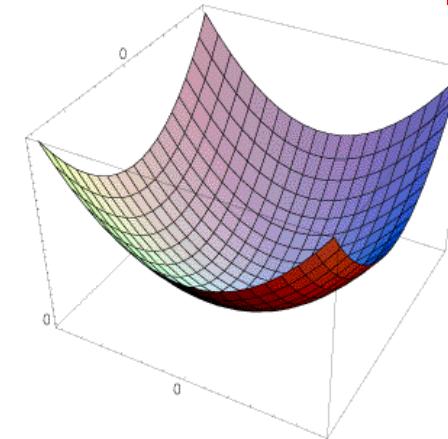
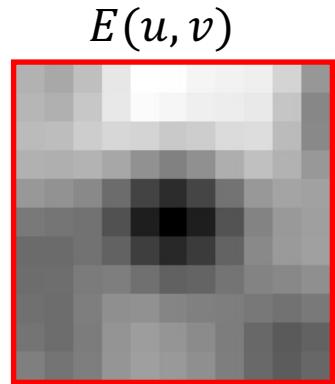


Rappel : DéTECTEUR de coins de Harris

- Approximation de l'unicité par l'autocorrélation locale
- Approximation de l'autocorrélation locale par le tenseur de structure H

$$H = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

- La similitude avec un coin est reliée aux valeurs propres de H
- Au lieu de calculer les valeurs propres directement, on utilise plutôt le déterminant et la trace de H



Rappel : DéTECTEUR de Harris

- Matrice du second moment

$$\mu(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

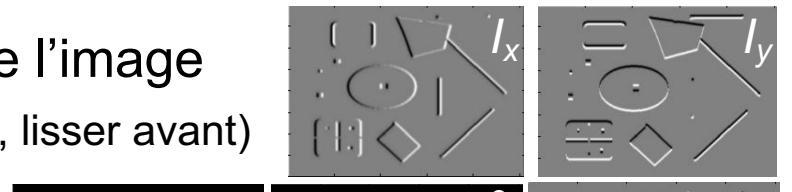
$$\det M = \lambda_1 \lambda_2$$

$$\text{trace } M = \lambda_1 + \lambda_2$$

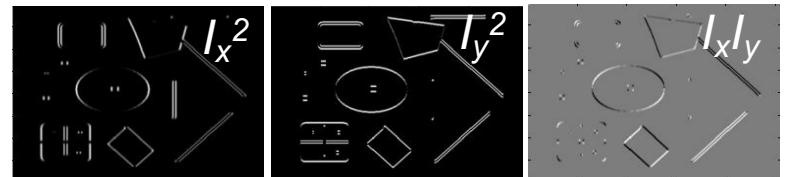
déterminant

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \rightarrow ad - cb$$

1. Dérivées de l'image
(optionnellement, lisser avant)



2. Carré des dérivés

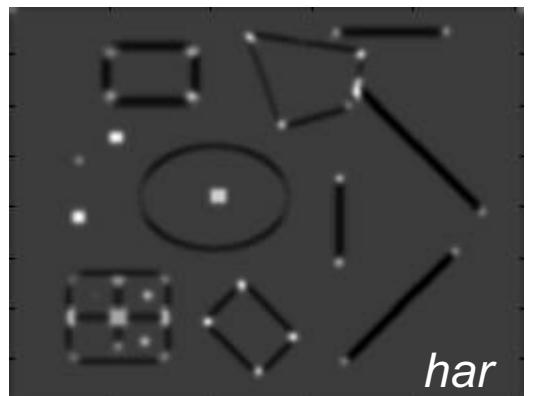


3. Filtre gaussien $g(\sigma_l)$



4. Fonction de similitude aux coins (*cornerness*)
– les deux valeurs propres sont grandes

5. Seuillage de C pour trouver les pics
6. Suppression des non-maximums



Quel est le niveau d'invariance des coins de Harris?

Chapitre 7, Partie 2 : Détection de caractéristiques

INF600F – Traitement d'images

Joël Lefebvre (UQÀM)

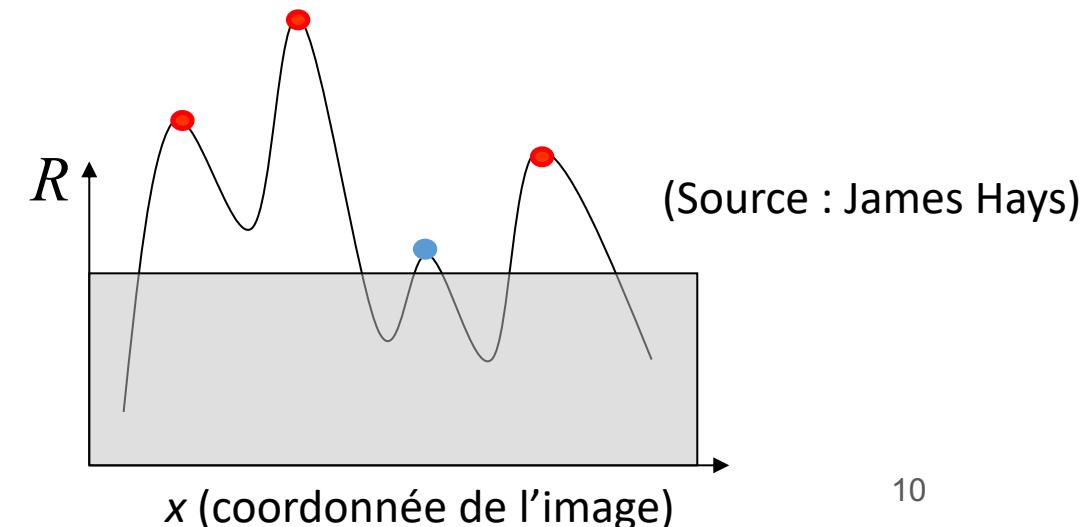
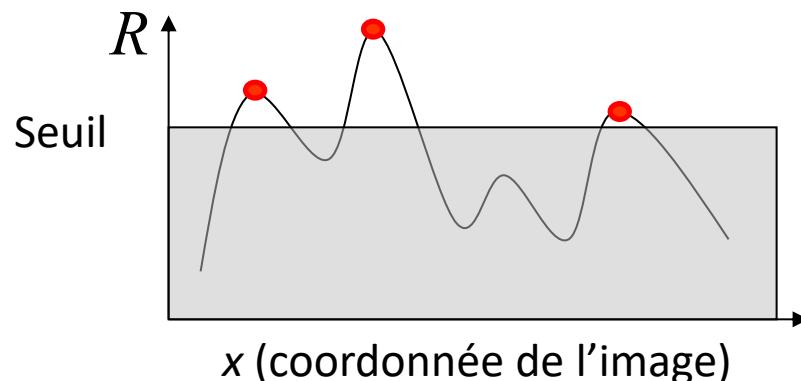
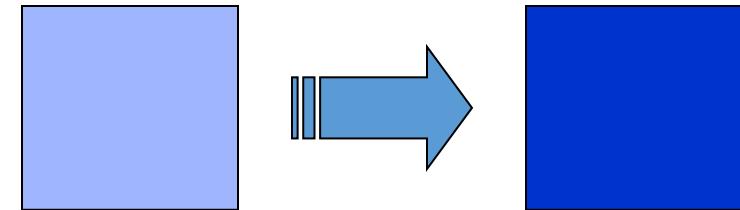
Rappel : Invariance et Covariance

- Est-ce que les positions des caractéristiques sont **invariantes** face à des **transformations photométriques** et **covariantes** avec les **transformations géométriques** ?
- **Invariance** : Si l'image est transformée, la position des coins ne change pas
- **Covariance** : Si on a 2 versions transformées de la même image (par exemple vue par une caméra à gauche et une caméra à droite), les points caractéristiques devraient être déplacés de la même façon que l'image.



Transformation photométrique

- Changement affine d'intensité : $I \rightarrow aI + b$
- Seulement les dérivées sont utilisées
- Invariance aux translations d'intensité : $I \rightarrow I + b$
- Mise à l'échelle de l'intensité : $I \rightarrow aI$
- **Partiellement invariant aux changements affines d'intensité**



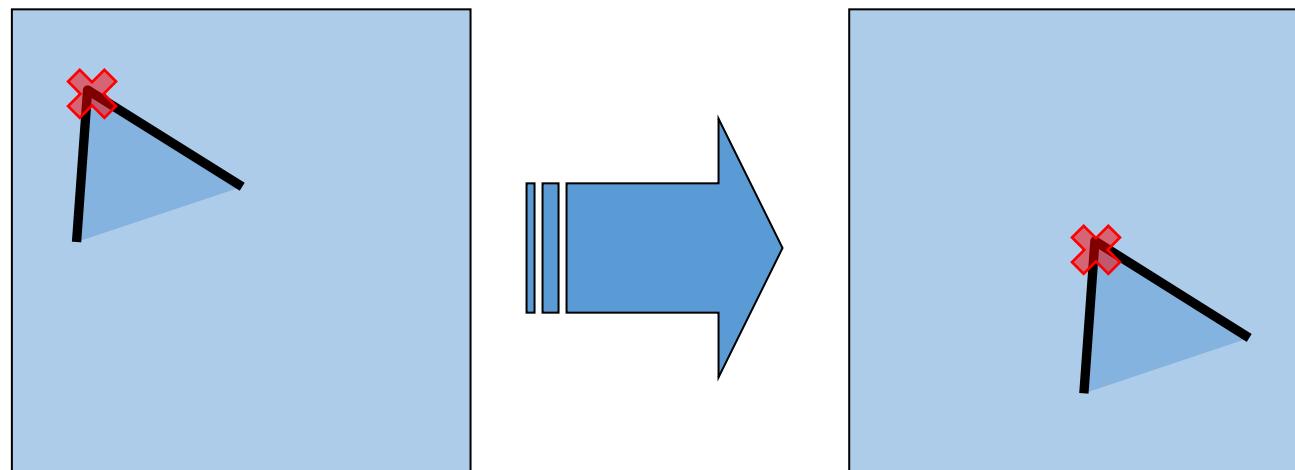
Transformation géométrique

Translation

- Les dérivées et la fonction de fenêtrage sont invariantes à la translation

$$H = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

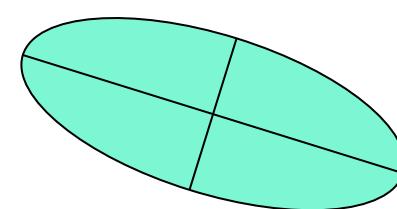
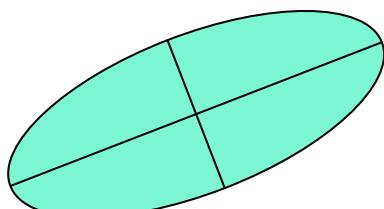
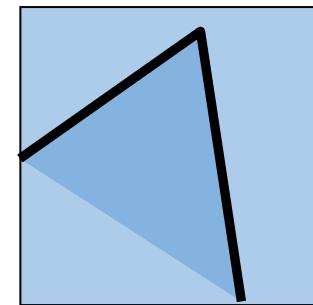
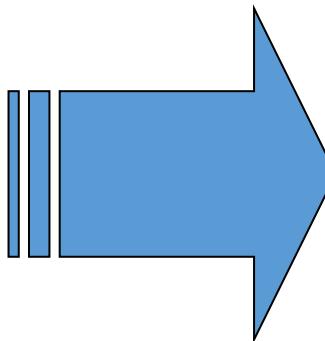
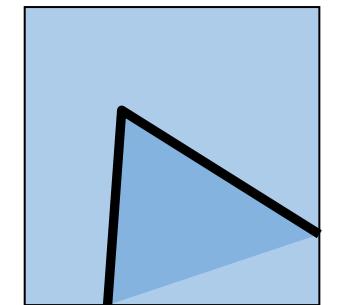
- La position des coins est covariante avec la translation



Transformation géométrique

Rotation

- L'orientation de l'ellipse du tenseur de structure est modifiée
- Sa forme (c.-à-d. valeurs/vecteurs propres) est la même
- **La position des coins est covariante avec la rotation**

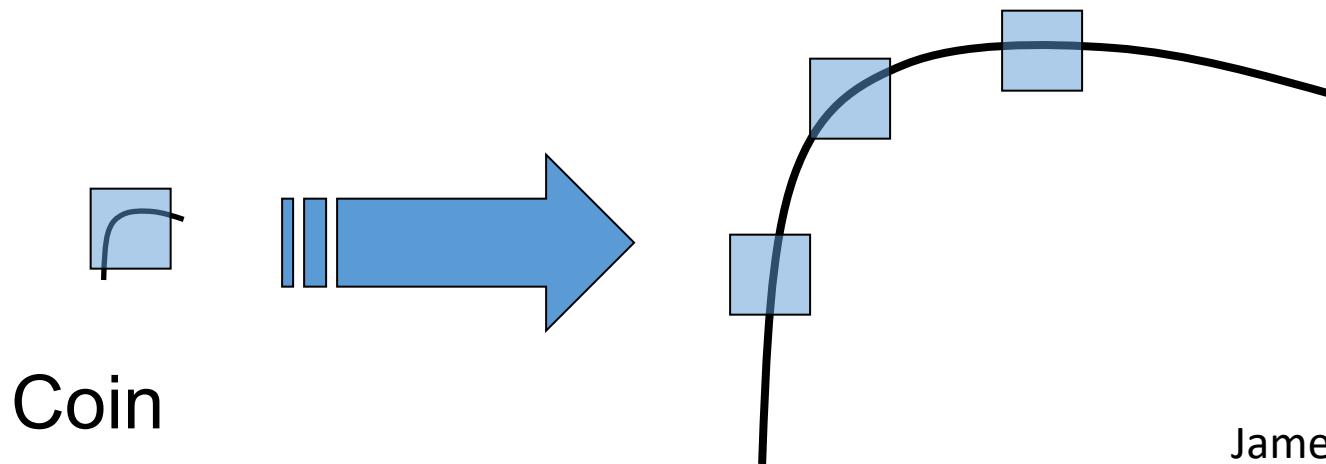


James Hays

Transformation géométrique

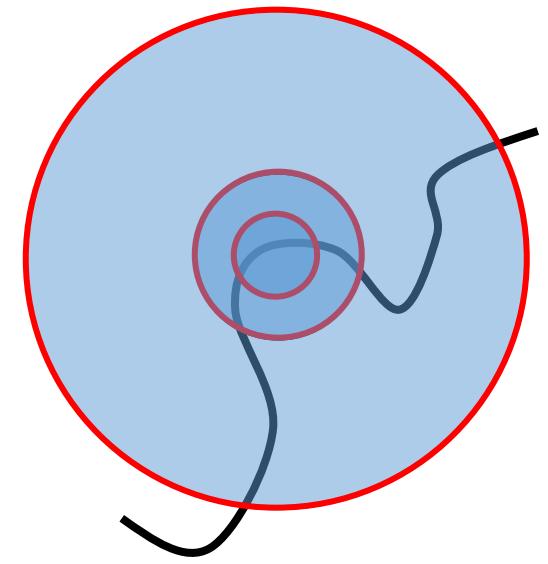
Mise à l'échelle

- Tous les points vont être classifiés erronément en tant que contour
- La position des coins n'est pas covariante avec l'échelle de l'objet
- Quelle est « l'échelle » d'un point caractéristique ?



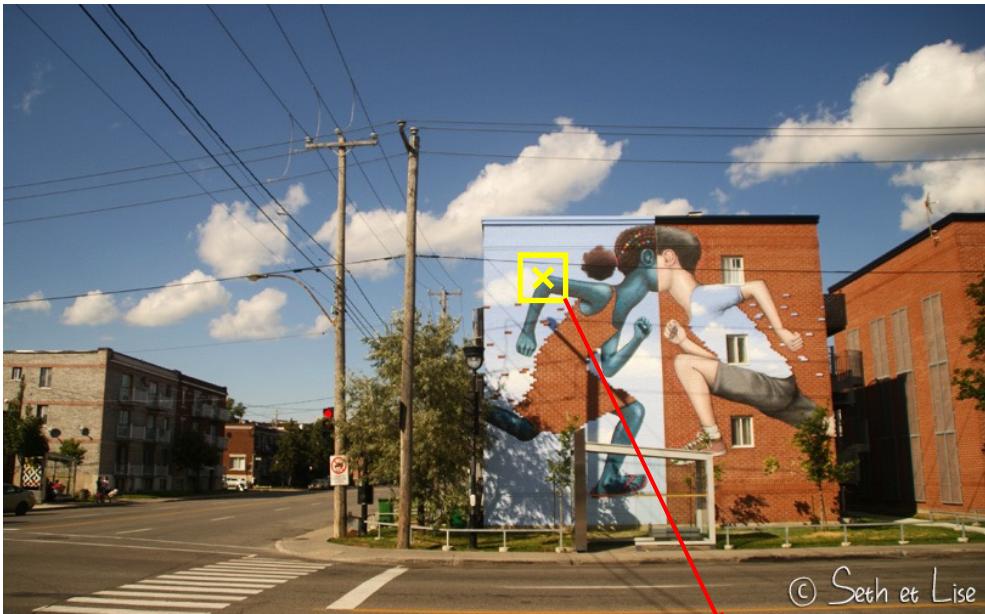
Détection et invariance d'échelle

- Supposons qu'on cherche un **coin**
- f est un détecteur de coin
- **Idée principale** : Trouver l'échelle qui donne le maximum local de f
- Maximum local autant pour la **position** que pour l'**échelle**
- Définition possible de f : **opérateur de Harris**



Détection automatique de l'échelle

- Comment trouver des **tailles** de sous-régions pour lesquelles la **réponse de f est égale** à chaque échelle ?
- Quelle serait une **bonne fonction caractéristique f** ?



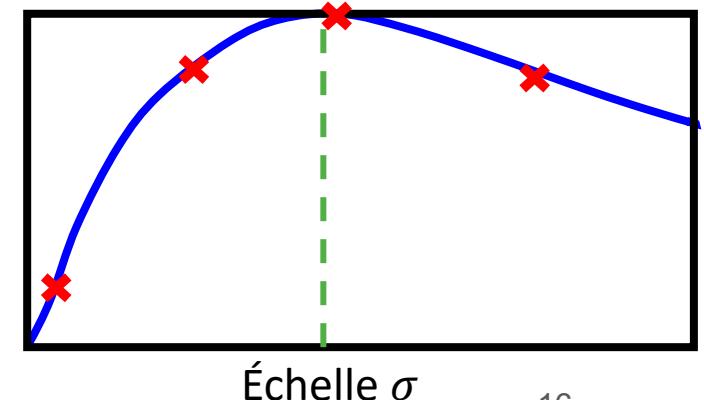
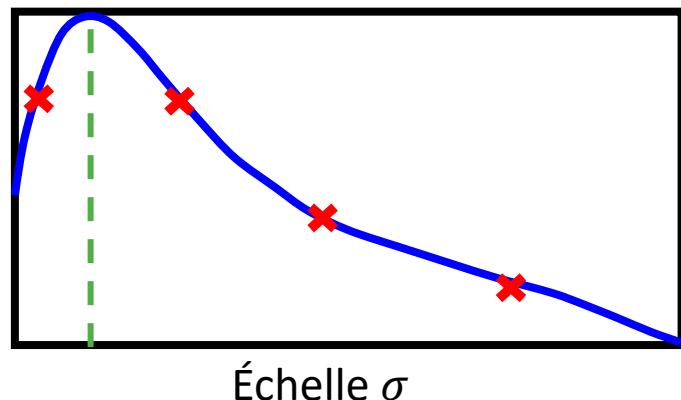
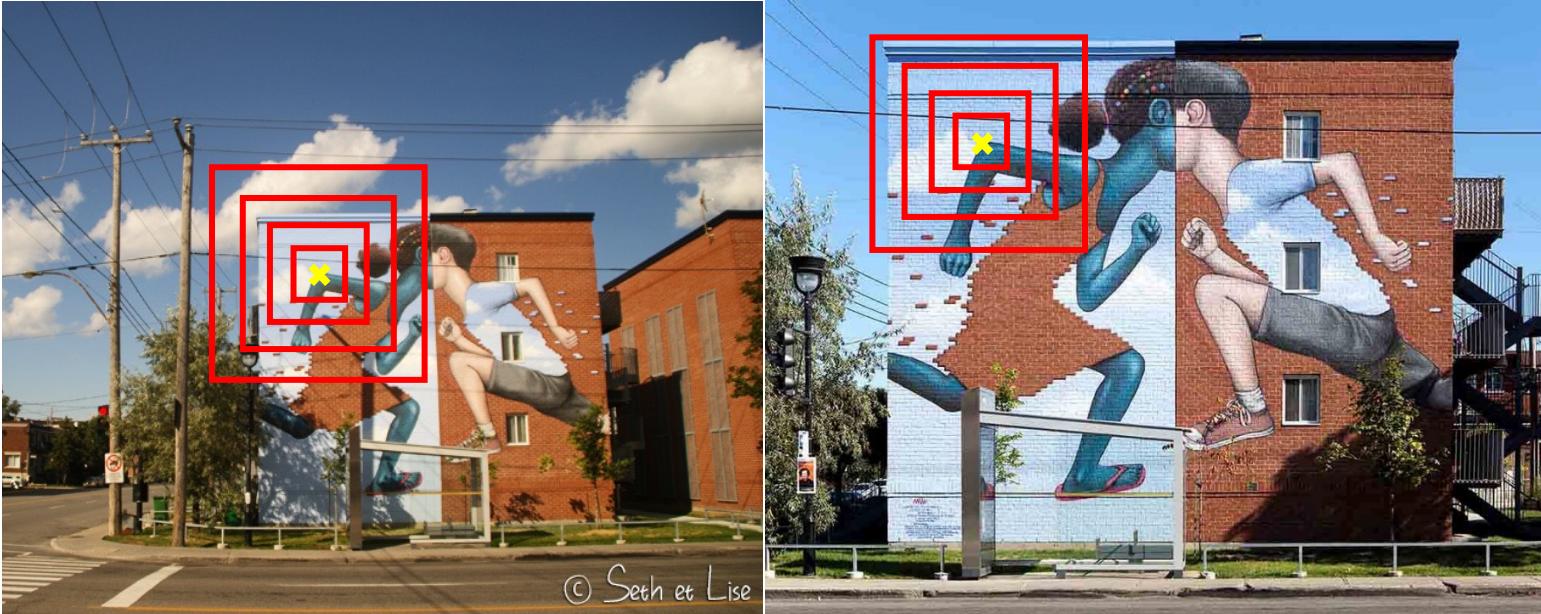
$$f(I_{i_1 \dots i_m}(x, \sigma)) = f(I_{i_1 \dots i_m}(x', \sigma'))$$



Murale « Brick Kidz »
de l'artiste Seth
Globepainter au coin
de Papineau / Jarry

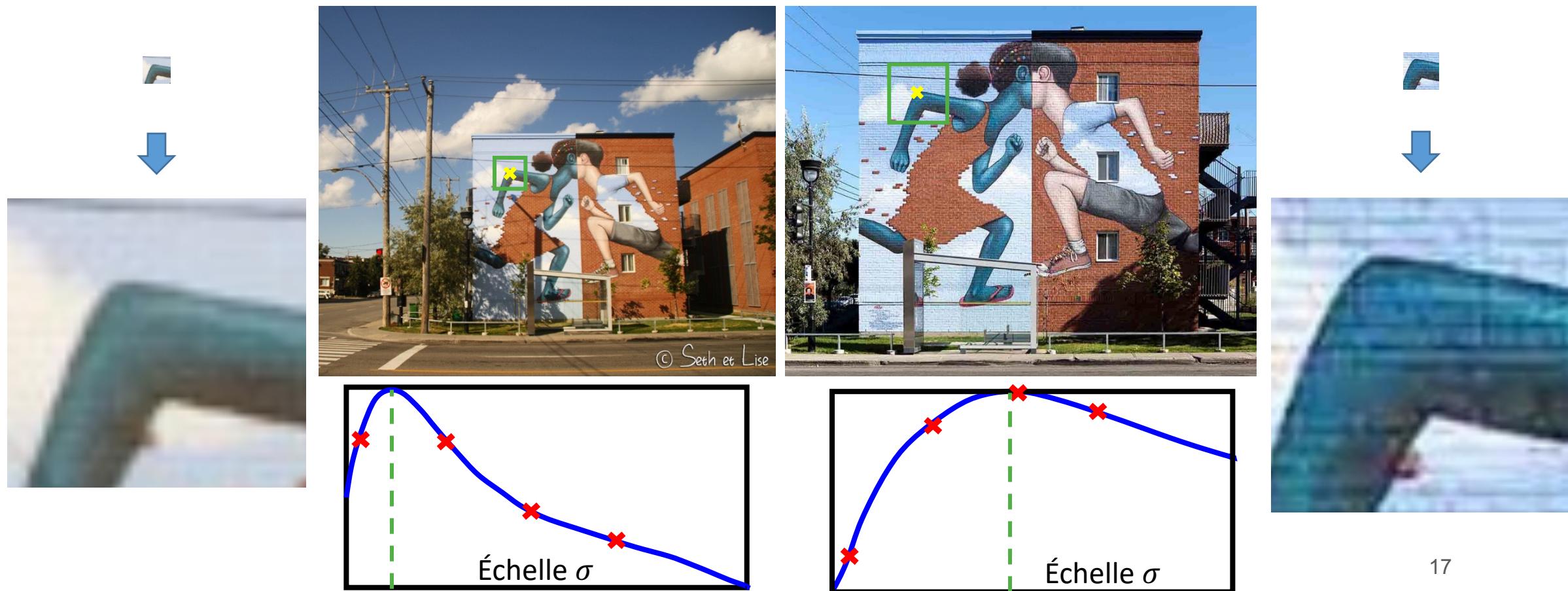
Détection automatique de l'échelle

- Réponses de la fonction f pour des échelles croissantes
- **En jaune** : Point caractéristique
- **En rouge** : Échelles échantillonnées
- **En bleu** : Signature multi-échelle de la fonction f
- **En vert** : Échelle correspondant à la réponse maximale
- Point caractéristique localisé **spatialement et à l'échelle** $p(x, y, \sigma)$



Détection automatique de l'échelle

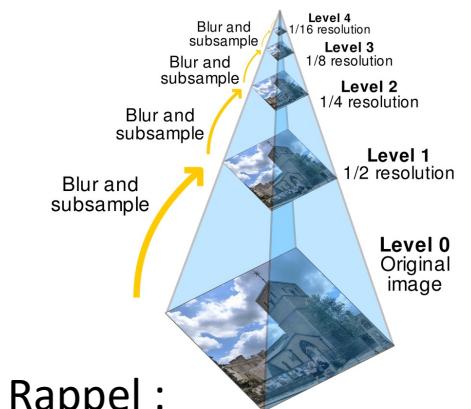
- **Normalisation** : Ré-échantillonnage à une échelle fixe



Détection automatique de l'échelle

Implémentation

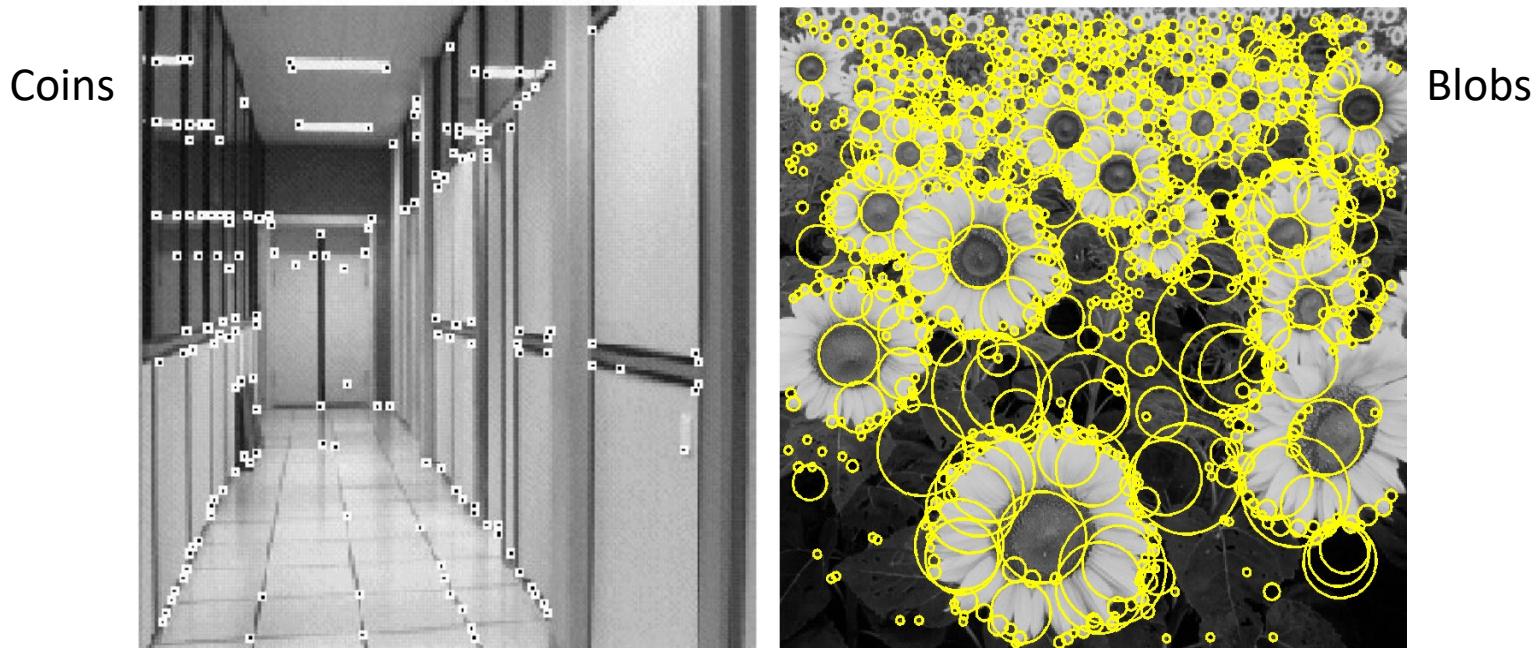
- Au lieu de calculer f pour des fenêtres de plus en plus grandes, on peut utiliser une taille de fenêtre fixe avec une décomposition en **pyramide de Gauss**



Rappel :

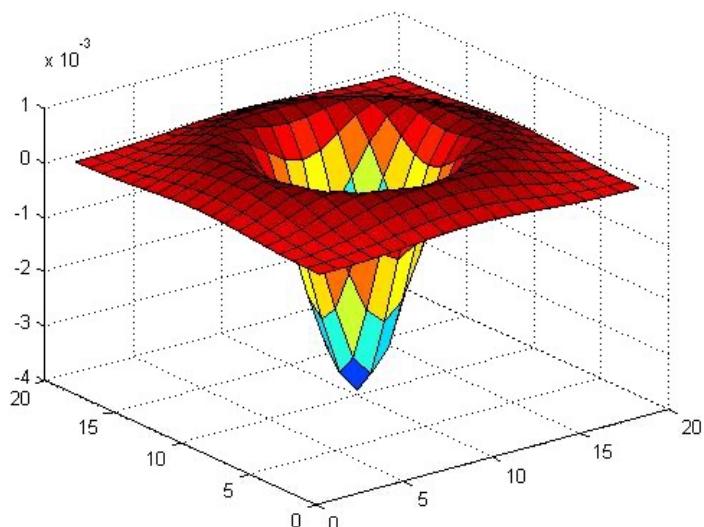
Point caractéristique : Taches (*Blobs*)

- Autre type de point caractéristique
- « Pic » ou « vallée » d'intensité
- Possède une certaine **taille caractéristique**

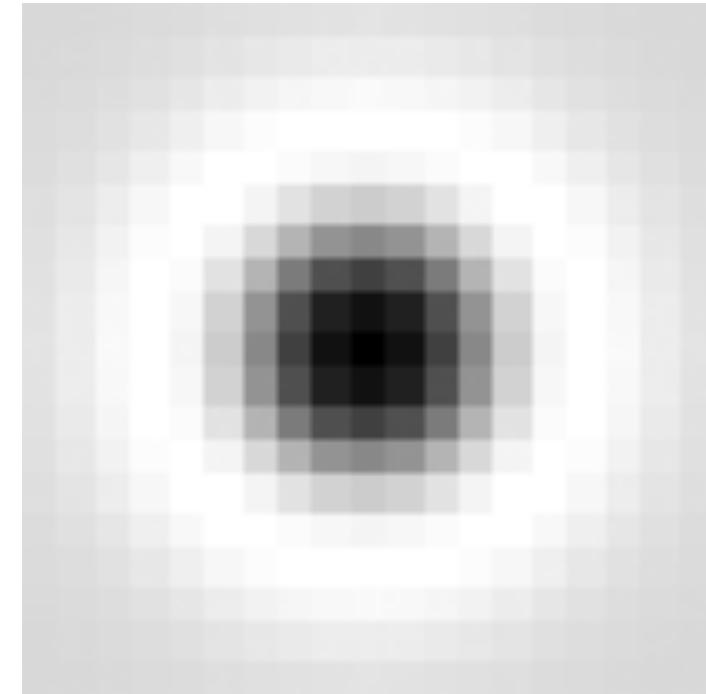


Autre définition commune pour f

- Détection des pics et vallées
- Le **Laplacien de la Gaussienne (LoG)**



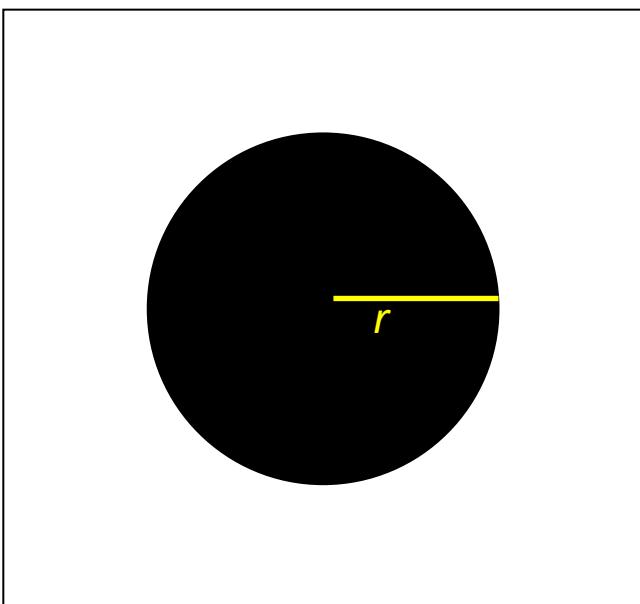
$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$



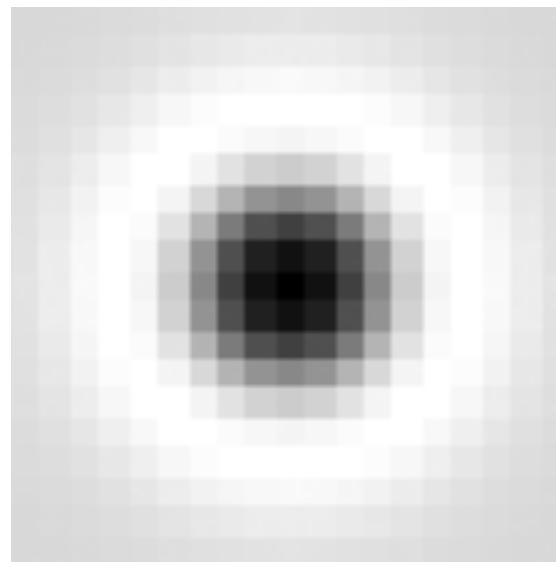
Rappel : LoG est très similaire à la **différence de gaussiennes (DoG)**, c.à.d une gaussienne moins une gaussienne légèrement plus petite.

Sélection de l'échelle

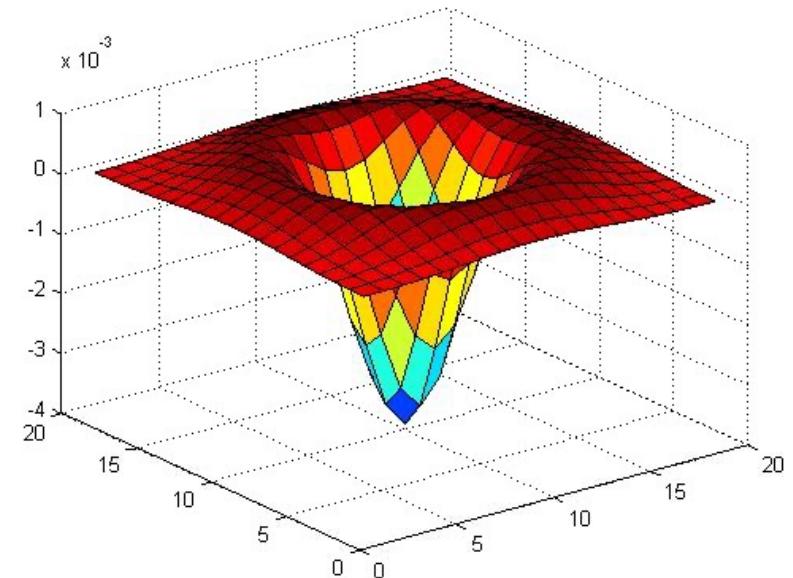
- À quelle échelle est-ce que le **Laplacien** a une réponse maximale pour un cercle binaire de rayon r ?



Image



Laplacien

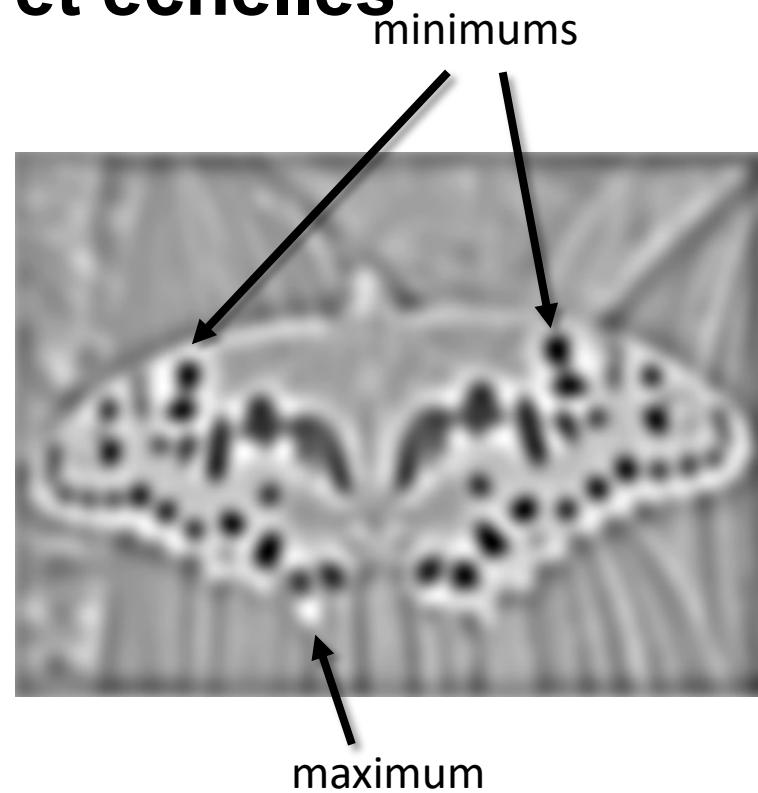


Laplacien de la gaussienne

- Détecteur de taches (**Blob**)
- Il faut trouver les **maximum et minimum locaux** de l'opérateur LoG, pour les dimensions **spatiales et échelles**

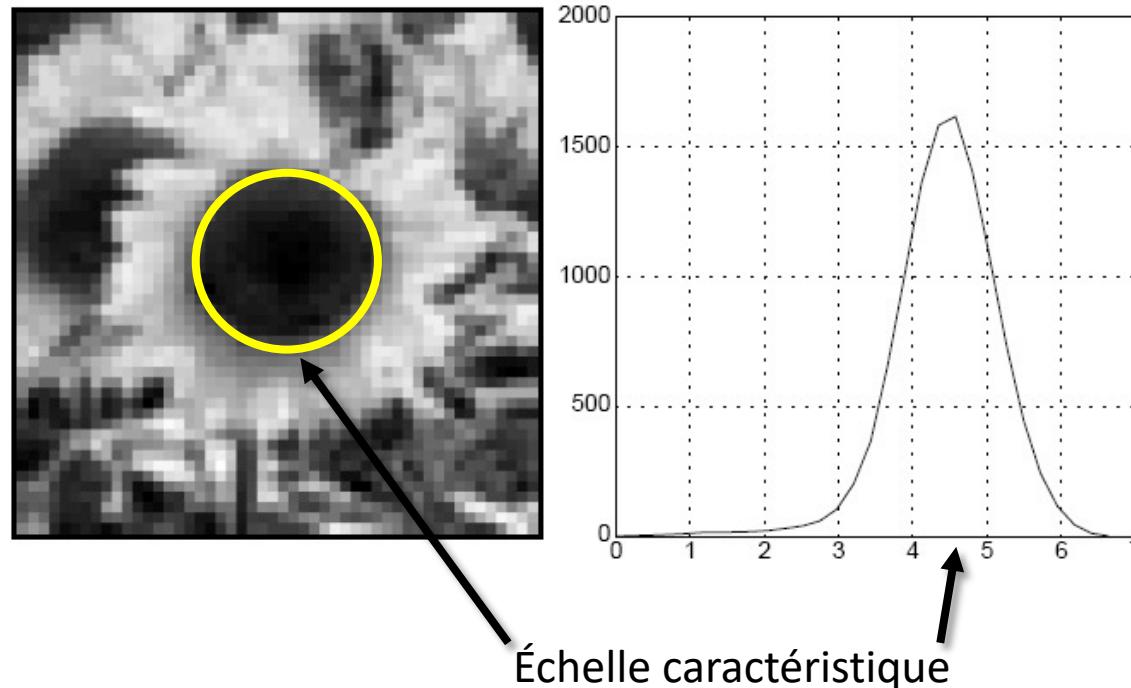


$$* \bullet =$$



Échelle caractéristique

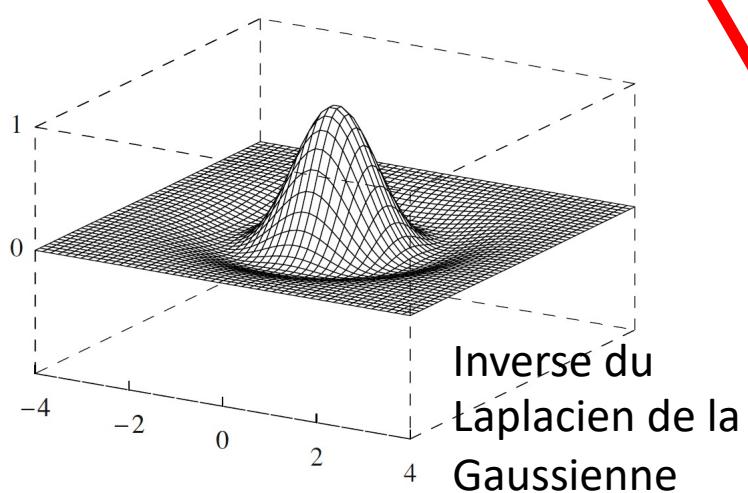
- On définit l'**échelle caractéristique** comme étant l'échelle qui produit la plus **forte réponse** pour l'opérateur Laplacien



Source : N. Snavely

T. Lindeberg (1998). ["Feature detection with automatic scale selection."](#)
International Journal of Computer Vision 30 (2): pp 77--116.

Détection du maximum local en 3D dans l'espace position-échelle



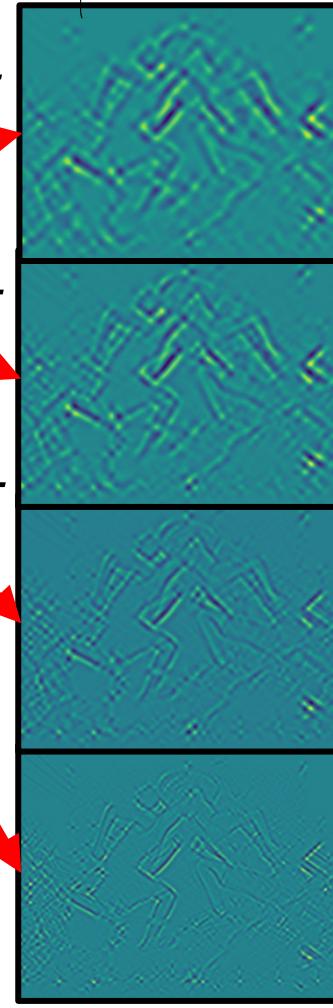
Scale-Space

4σ

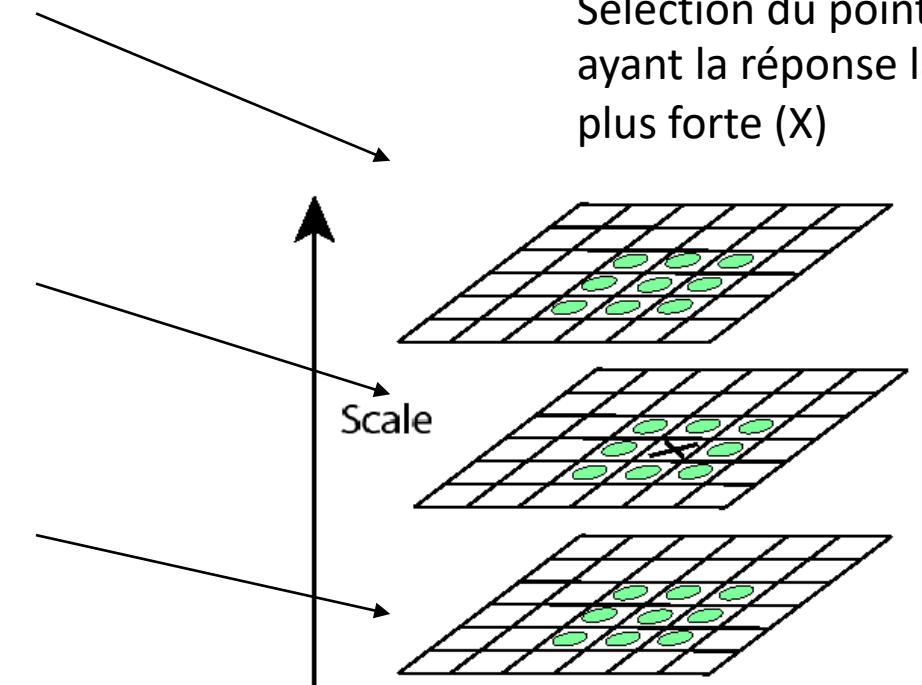
3σ

2σ

σ



⇒ Liste des positions
(x, y, s)



Rappel : Similitude en DoG et LoG

- Fonctions pour déterminer l'échelle :

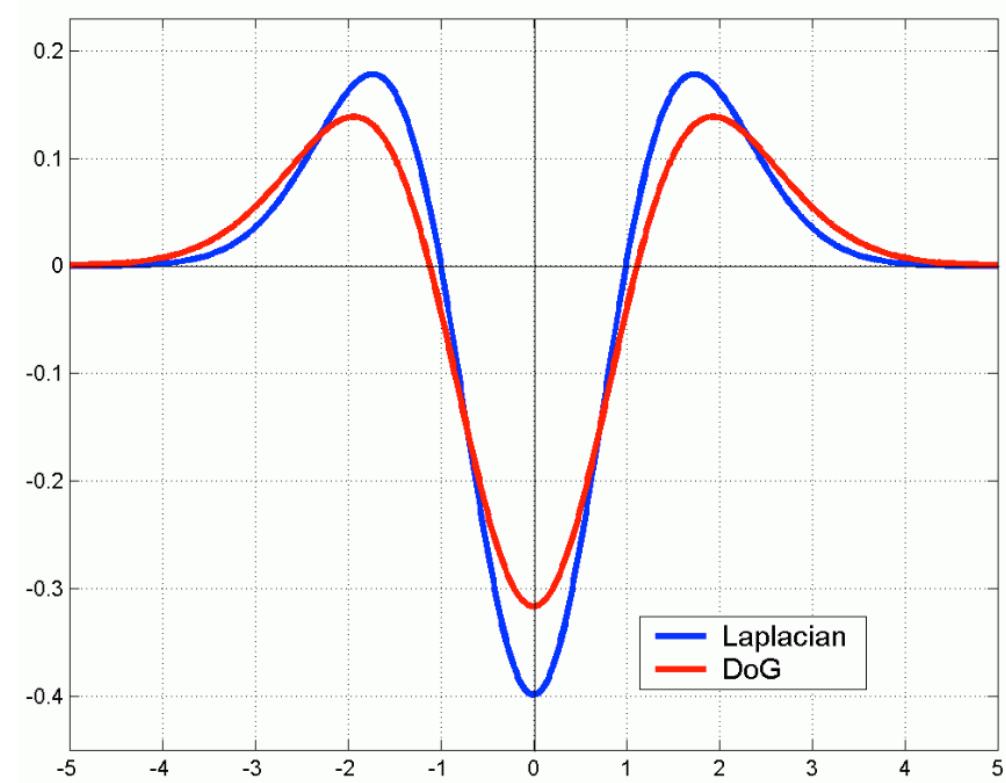
$$f = \text{Noyau} * \text{Image}$$

- Noyau Laplacien (LoG): $\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$

- Noyau « Différence de gaussiennes » :
$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

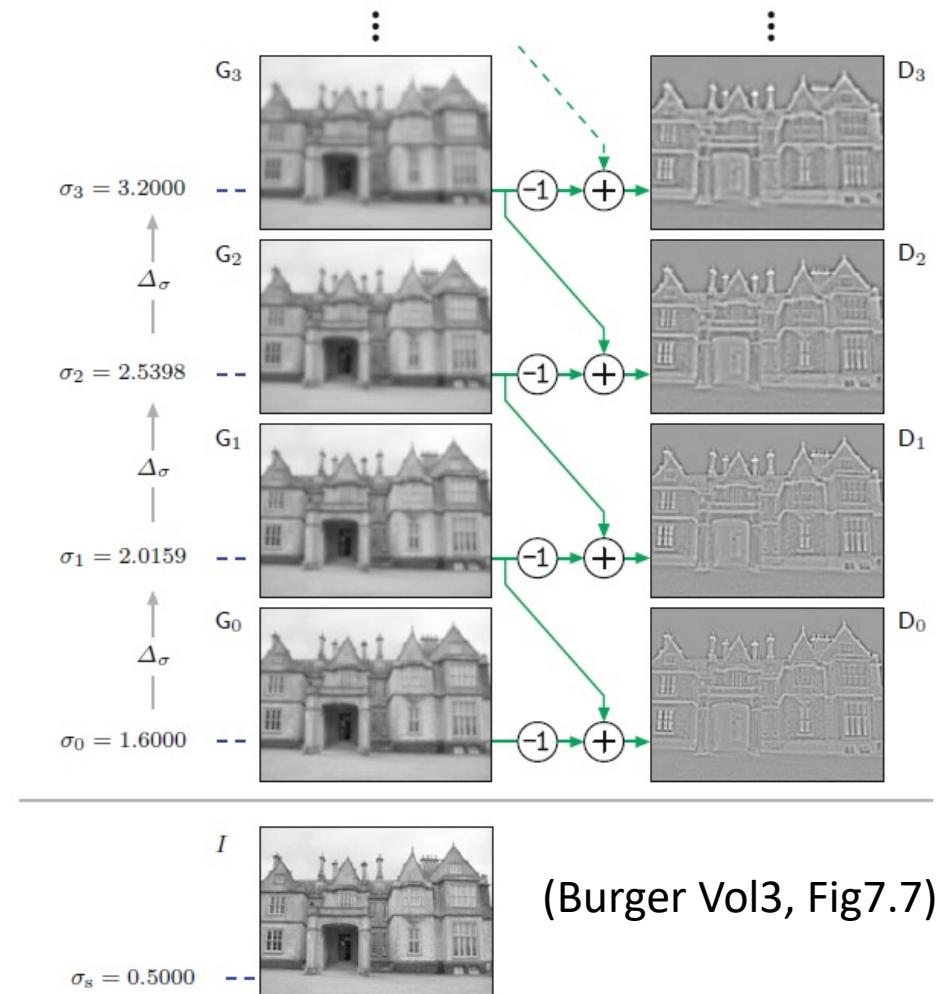
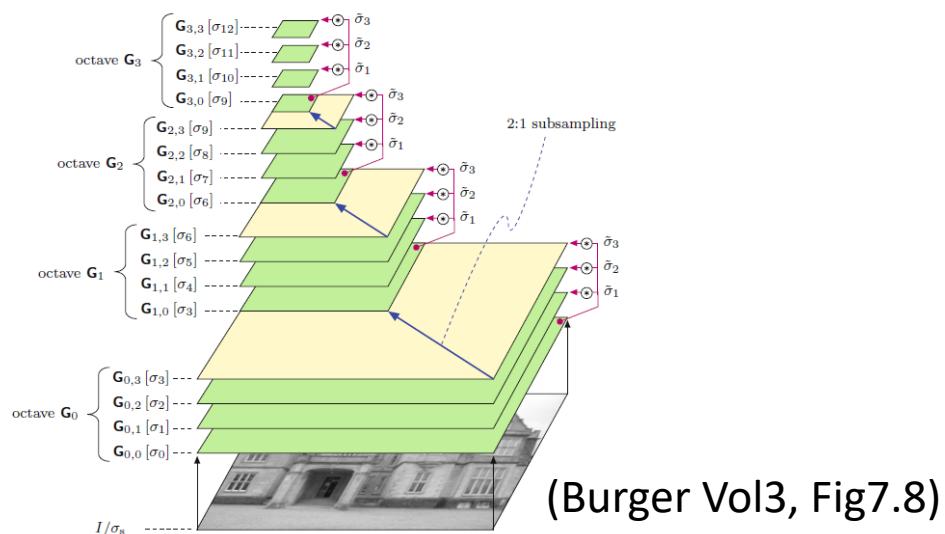
Avec $G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$

Note: Les opérateurs LoG et DoG sont équivariants avec la rotation

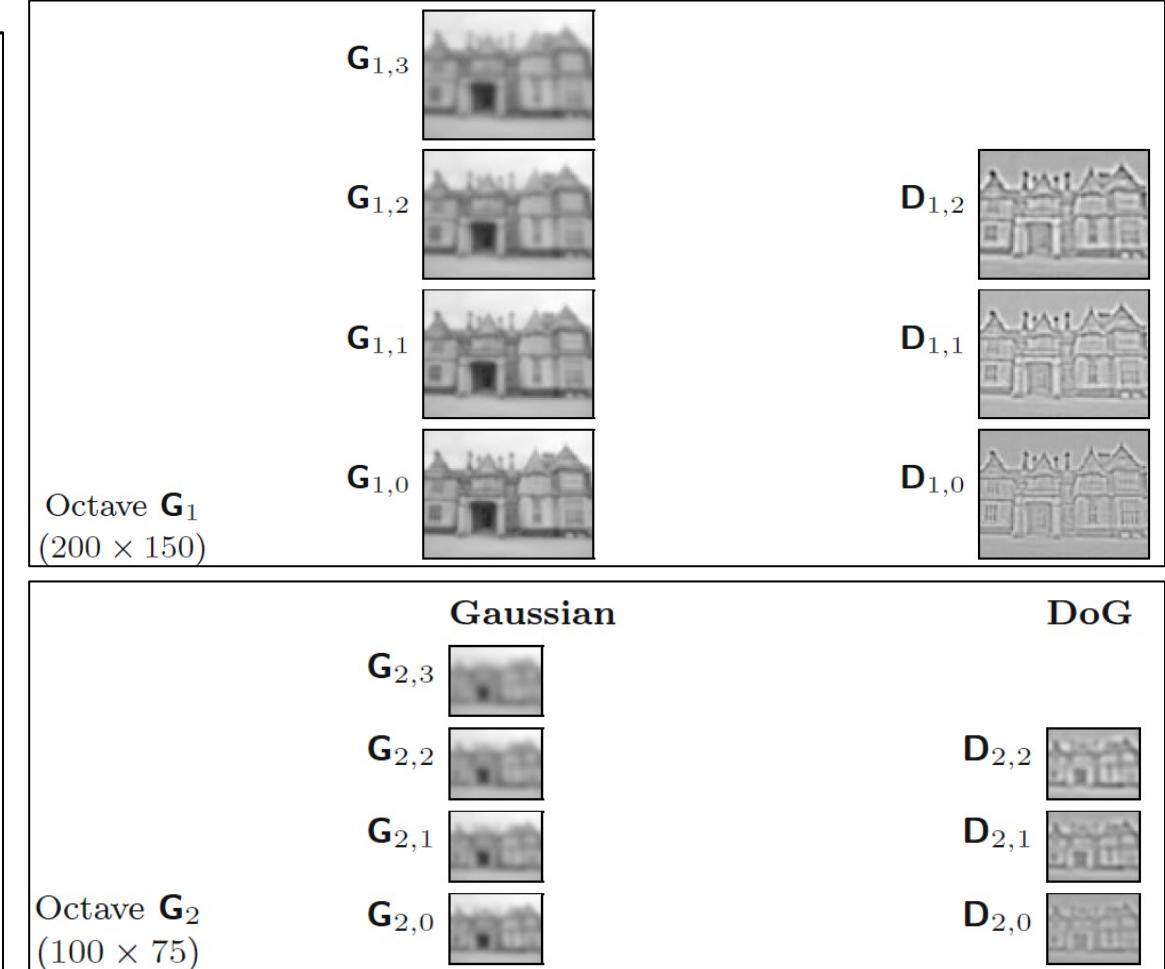
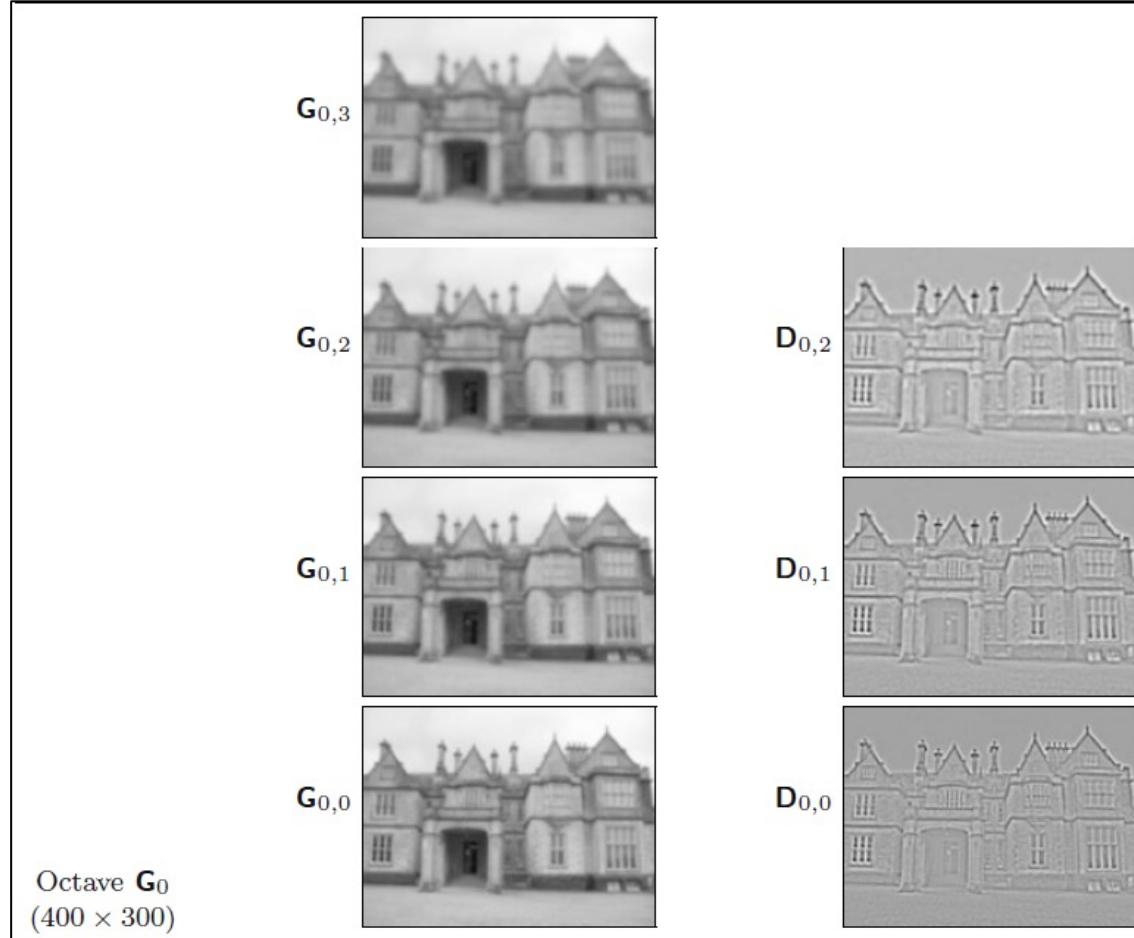


En pratique : Pyramide de Gauss

- Le Laplacien de la Gaussienne (LoG) peut être approximée par une différence de Gaussienne (DoG)
- Le contenu fréquentiel est réduit par 2 à chaque octave, on peut donc décimer d'un facteur 2
- Octave : Échelle pour laquelle σ double

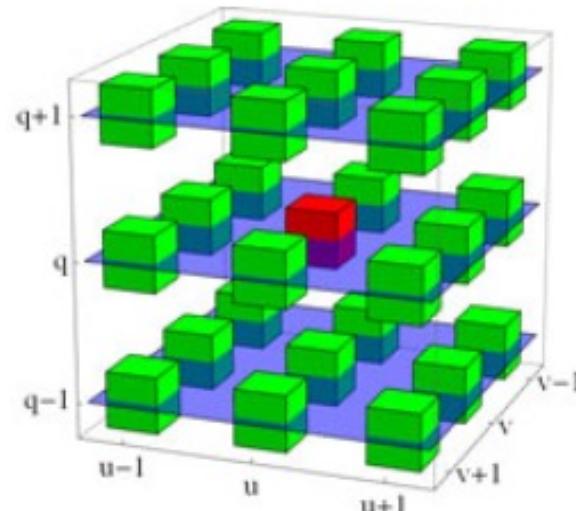


Exemple : Espace échelle gaussien et DOG hiérarchique

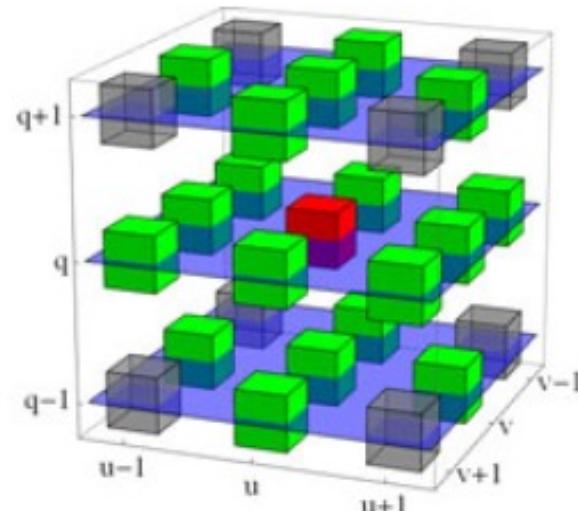


Détection des extrêmes locaux

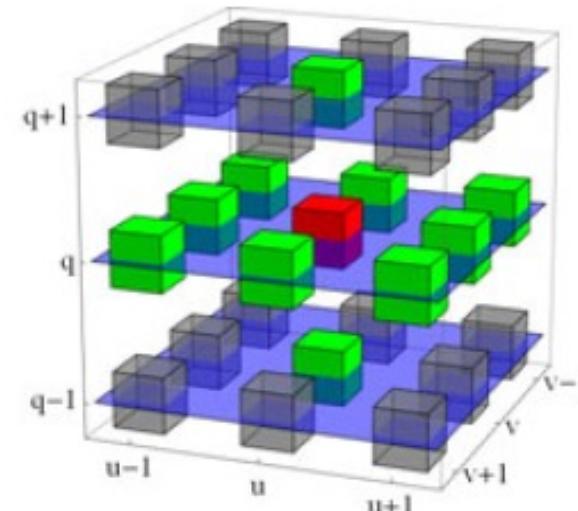
- Détection d'un **extremum local** dans l'espace échelle DoG
- **Extremum** : tous pixels du voisinage on une intensité plus petite (maximum) ou plus grande (minimum) que le pixel central
- On doit définir **un voisinage 3D**, les dimensions sont (x, y, σ)



(a) 26-neighborhood



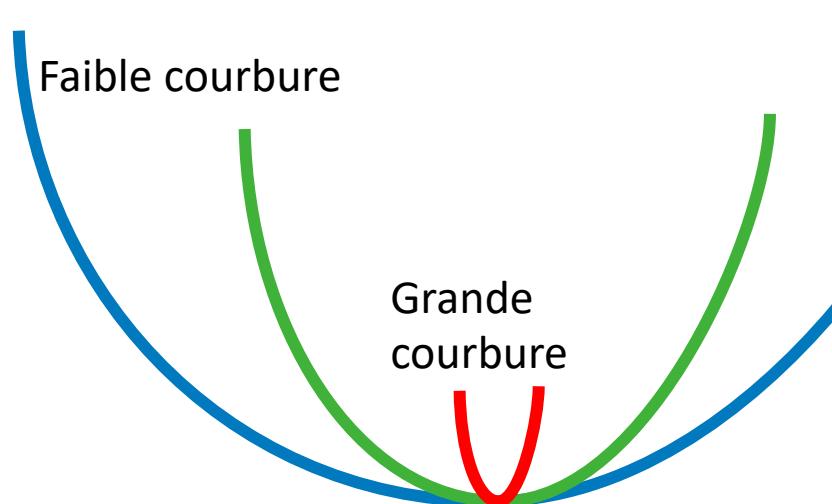
(b) 18-neighborhood



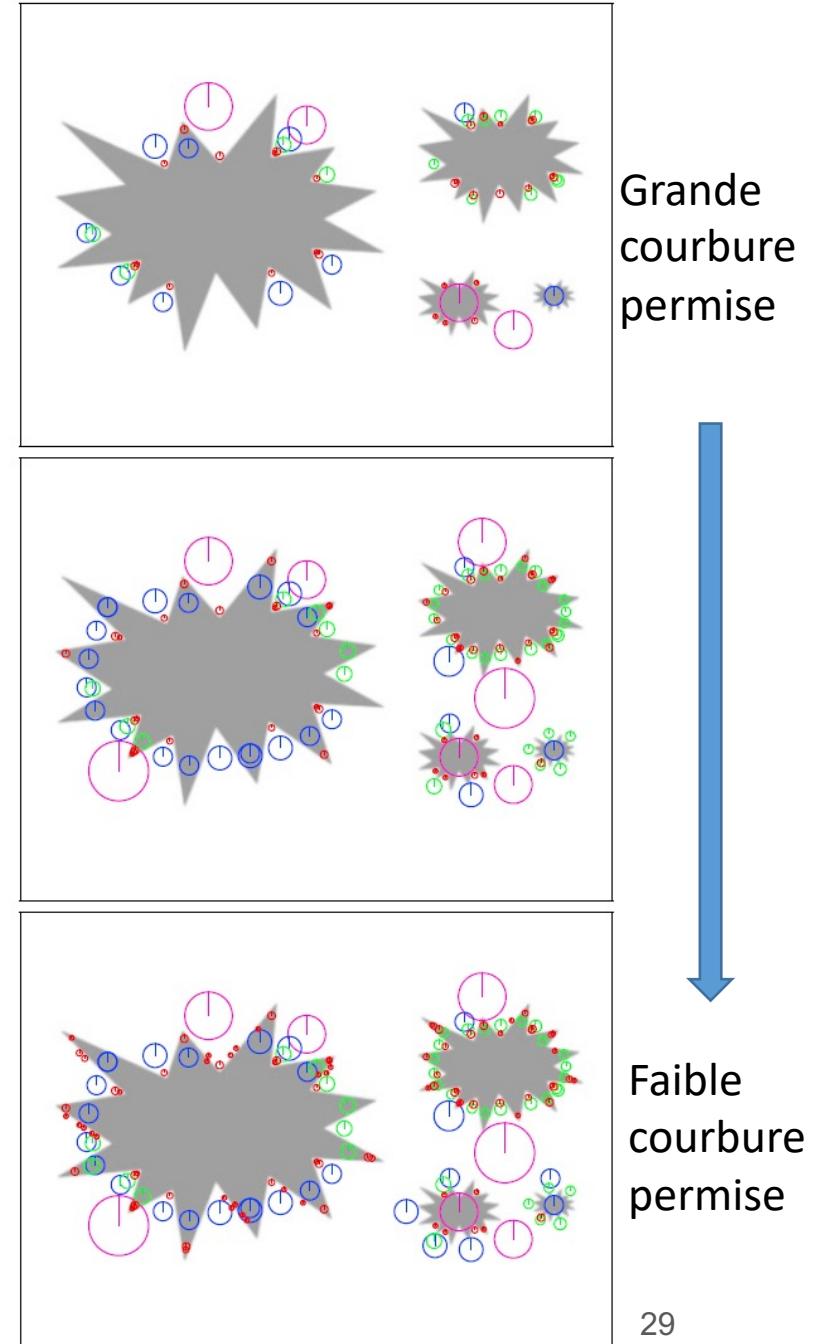
(c) 10-neighborhood

Détection des blobs : Étapes supplémentaires

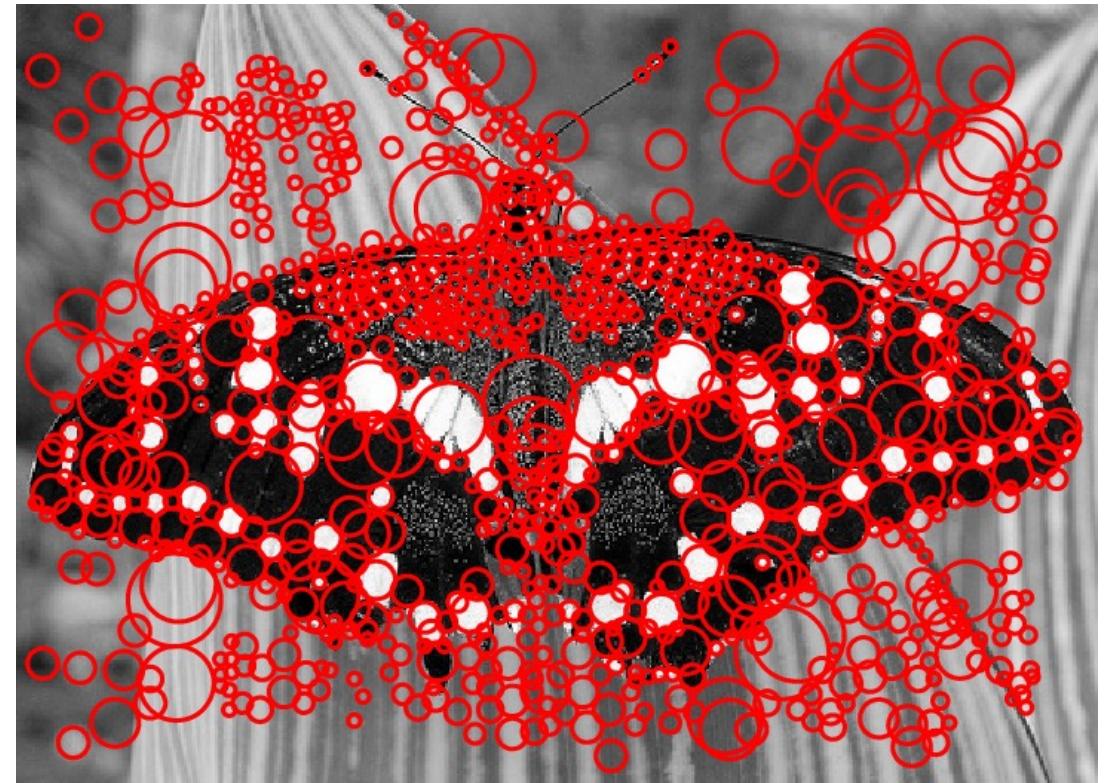
- Détection des extreums locaux
- Transformation des positions discrètes en positions continues
- Suppression des contours



Exemple de suppression des points caractéristiques sur les contours
(basé sur la courbure locale).
Source : Burger Vol3, Fig7.17



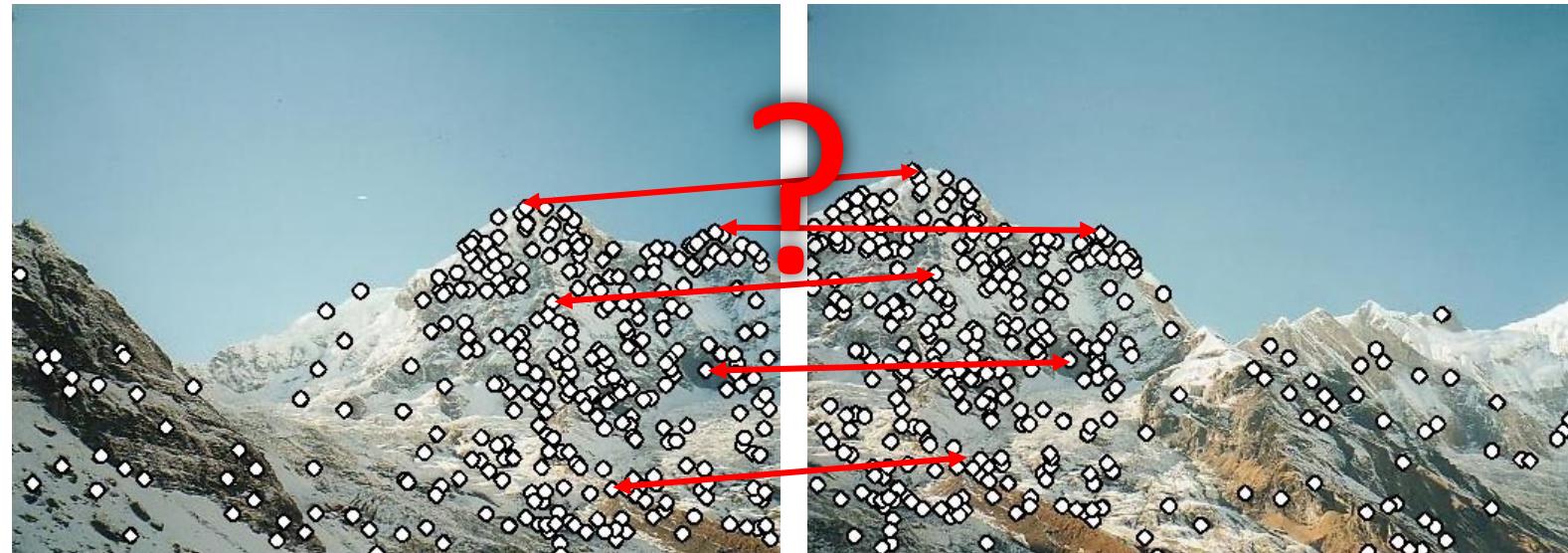
Exemple : Détection de taches dans l'espace spatial / échelle



$\sigma = 11.9912$

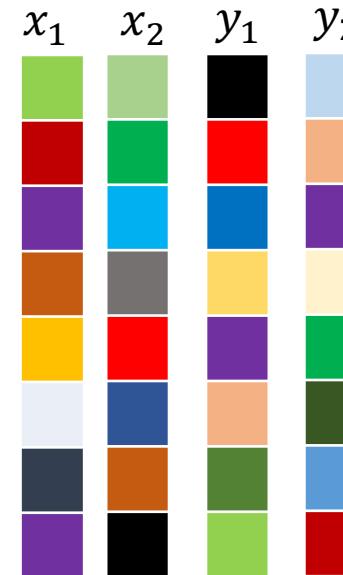
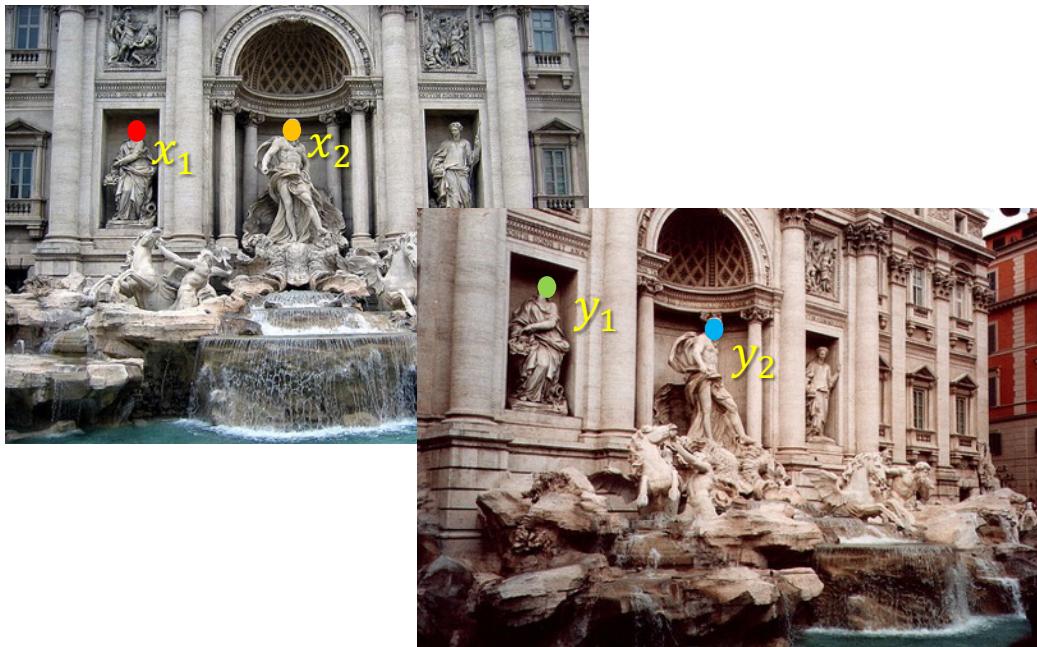
Descripteurs de caractéristiques

- On sait comment détecter de **bons points caractéristiques**
- Prochaine question : **Comment les associer ?**
- **Solution** : Établir un *descripteur* pour chaque point, puis trouver les descripteurs similaires entre les deux images



Descripteur & Pairage de points caractéristiques

- Créer des descripteurs pour chaque point caractéristique
- Mesurer la distance $d(x_i, y_j)$ (ou similarité) entre chaque paire de descripteurs



x_1	x_2	y_1	y_2	y_1	y_2
x_1				$d(x_1, y_1)$	$d(x_1, y_2)$
	x_2			$d(x_2, y_1)$	$d(x_2, y_2)$

Invariance vs discriminabilité

- **Invariance**
 - Le descripteur ne devrait pas changer même si l'image est transformée
- **Discriminabilité**
 - Le descripteur devrait être fortement unique pour chaque point

Rappel : Transformations d'images

- Géométrique

- Translation
- Rotation
- Échelle



Rotation

Échelle



- Photométrique

- Changement d'intensité



Changement
d'intensité



Descripteurs invariants

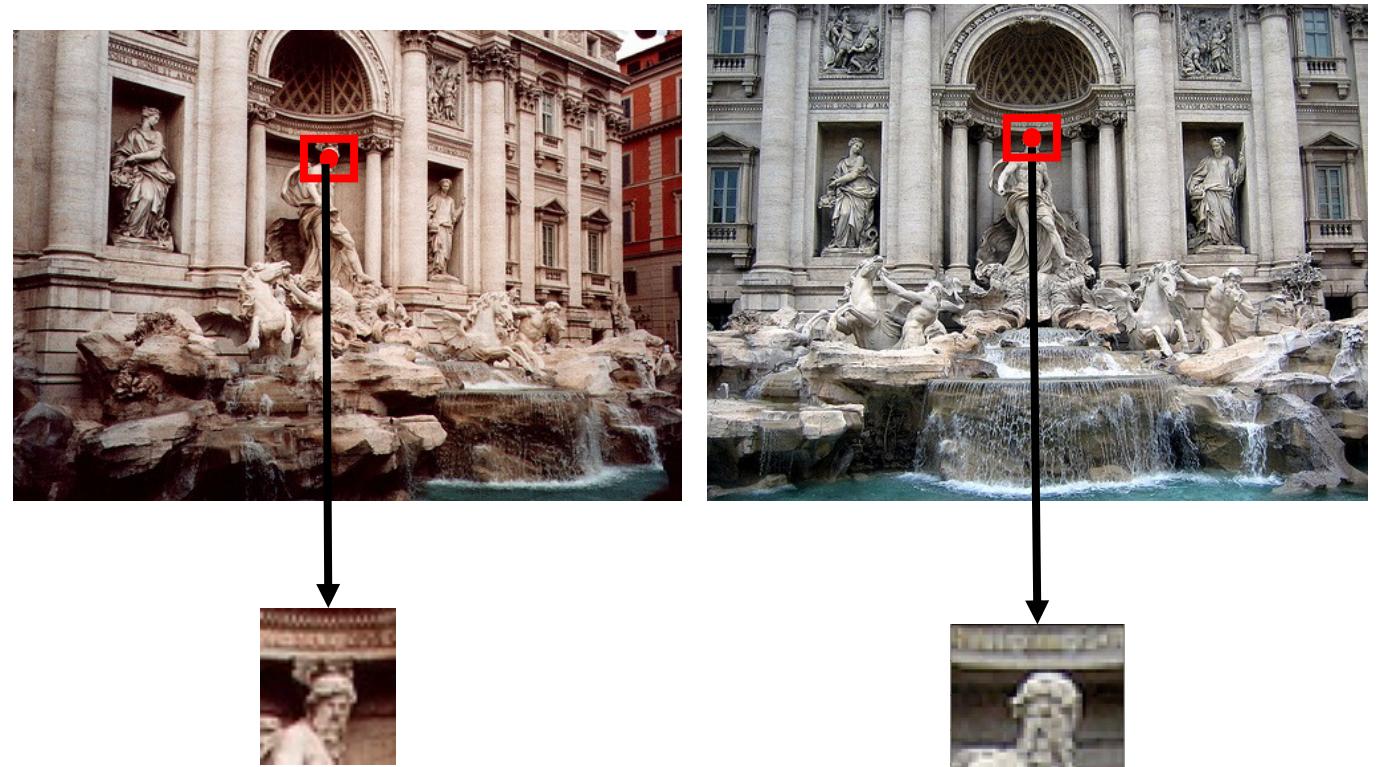
- On a étudié des **détecteurs** invariants et équivariants
- La plupart des **descripteurs** sont aussi conçus pour être **invariants** aux
 - Translation, rotation 2D, mise à l'échelle
- Ils peuvent aussi être **partiellement invariants**
 - Rotations 3D
 - Transformations affines limitées
 - Transformations limitées d'illumination / contraste

Descripteur : Point vs Sous-région

Un seul pixel



Sous-région autour du point caractéristique

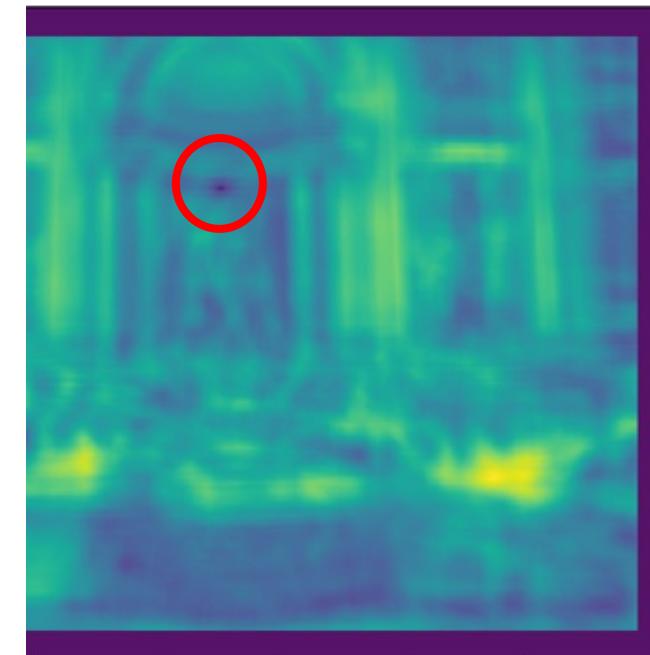
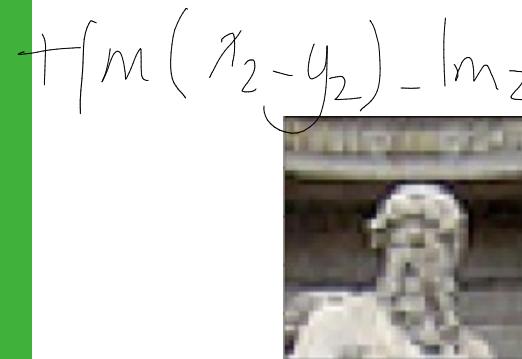


Distance entre les descripteurs ?
2 façons

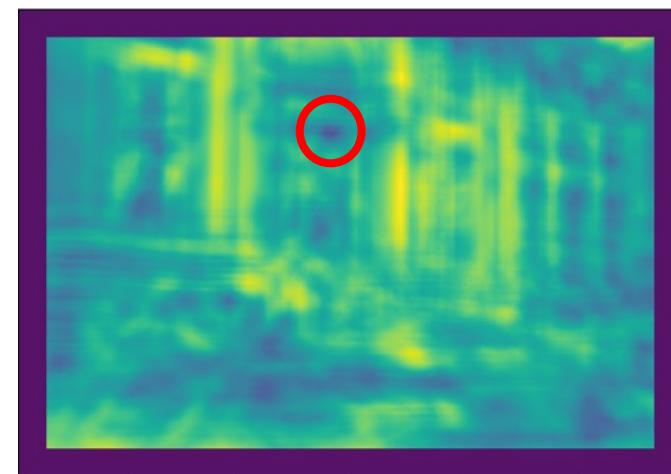
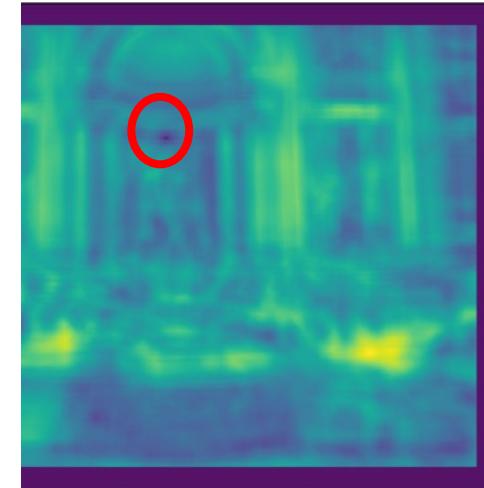
Somme des différences carrées (SSD)

- Utiliser en tant que descripteur la sous-région en entier
- Associer les descripteurs avec la distance Euclidienne

$$d(x, y) = \|x - y\|^2$$
$$\|m_1(x_1, y_1) - m_2(x_1, y_1)\|^2$$



Exemple : Association entre 2 images



Exemple : Correspondance par SSD (Pont)

- Pont Jacques-Cartier (Python et OpenCV)

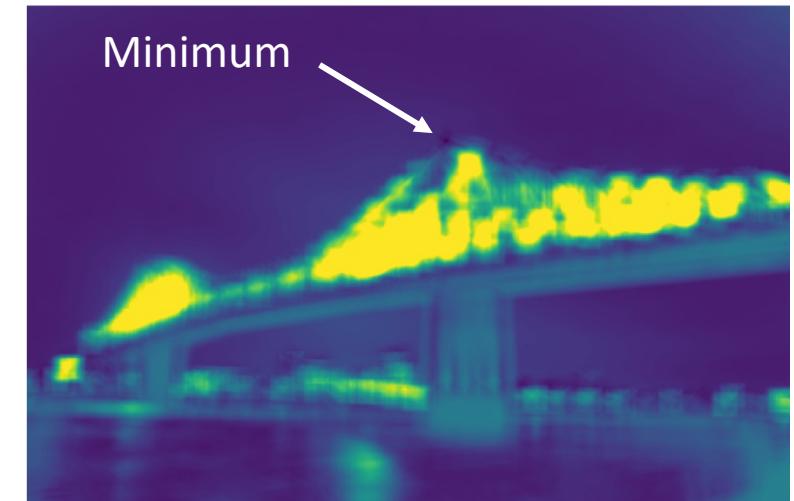
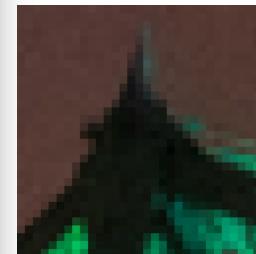
```
import cv2
from matplotlib.patches import Rectangle

img1 = img_mystere.copy()
img2 = img_list[7]

r0 = 175; c0 = 545
size = 32

patch = img1[r0:r0+size, c0:c0+size]

img_gray = img1.mean(axis=2).astype(np.float32)
patch_gray = patch.mean(axis=2).astype(np.float32)
match = cv2.matchTemplate(img_gray, patch_gray, cv2.TM_SQDIFF)
```

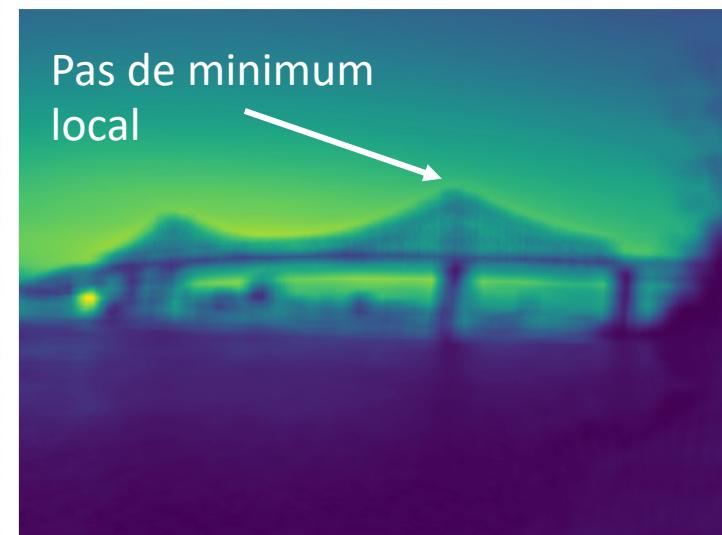


Exemple : Performances SSD Changements de couleur



Image 2

SSD



Corrélation croisée normalisée (NCC)

- Invariant à certains changements d'illumination, de couleur et d'intensité des pixels
- **Modèle** d'augmentation du contraste / luminosité

$$I' = \alpha I + \beta$$

- Il faut d'abord **normaliser** les descripteurs

- Soustraction de la **moyenne d'intensité** $\langle x \rangle$: invariance face à β
- Division par la **norme vectorielle** $\|x\|$: invariance face à α

$$x' = x - \langle x \rangle$$

$$x'' = \frac{x'}{\|x'\|}$$

$\xrightarrow{\frac{1 - \langle x \rangle}{\text{Var}(1 - \langle x \rangle)}}$
 $\mu \approx 0$
 $\sigma^2 \approx 1$

- **Similarité** = $x'' \cdot y''$

Exemple : Correspondance par NCC (Pont)

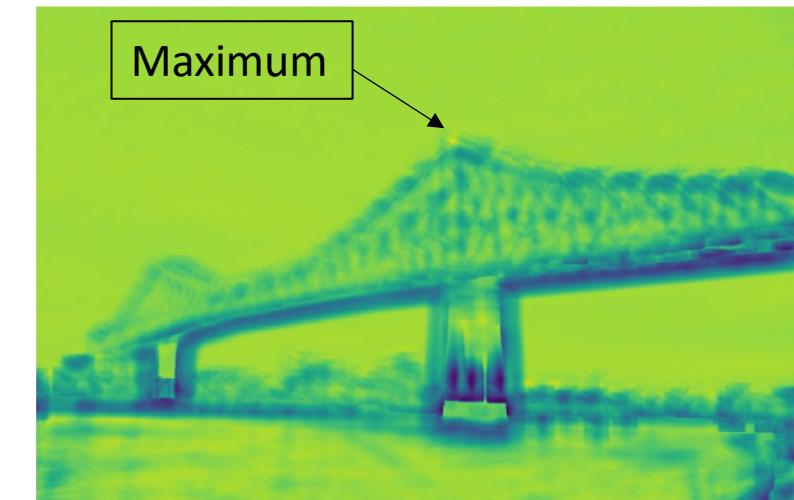
```
# Comparaison NCC
import cv2
from matplotlib.patches import Rectangle

img1 = img_mystere.copy()
img2 = img_list[7]

r0 = 175; c0 = 545
size = 32

patch = img1[r0:r0+size, c0:c0+size]

img_gray = img1.mean(axis=2).astype(np.float32)
patch_gray = patch.mean(axis=2).astype(np.float32)
match = cv2.matchTemplate(img_gray, patch_gray, cv2.TM_CCORR_NORMED)
```



Exemple : Performance SSD vs NCC

Changement de couleur



Correspondances de base

- **Descripteur** : Sous-région d'images autour d'un point caractéristique
- **Fonction de similarité** : NCC
- **Invariances**
 - Transformation photométrique ? (Oui, si décrit par $I' = \alpha I + \beta$)
 - Translation ? (Oui)
 - Rotation ?
 - Échelle ?

Invariance à la rotation pour les descripteurs de caractéristiques

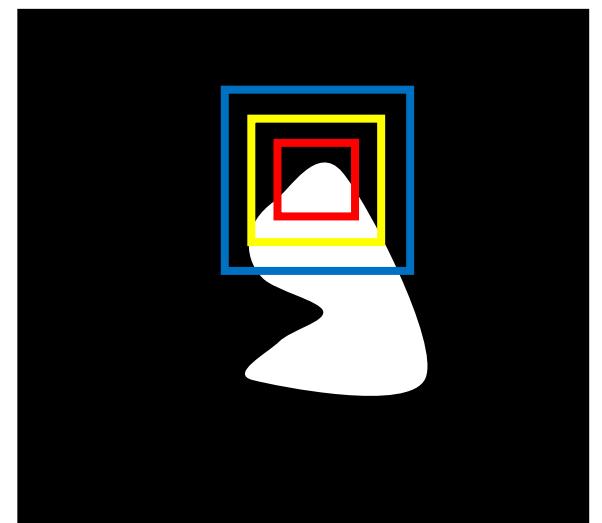
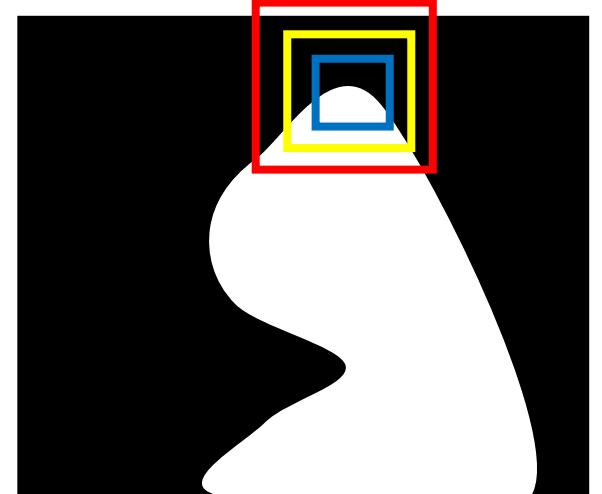
- Trouver l'**orientation dominante** d'une sous-région de l'image
- **Option 1** : La direction donnée par x_{max} , le vecteur propre de H correspondant à λ_{max} (la valeur propre la plus grande)
- **Option 2** : L'orientation du gradient
- Ensuite, il faut **pivoter** la sous-région pour l'aligner avec l'orientation dominante



Figure by Matthew Brown

Invariance à l'échelle pour les descripteurs

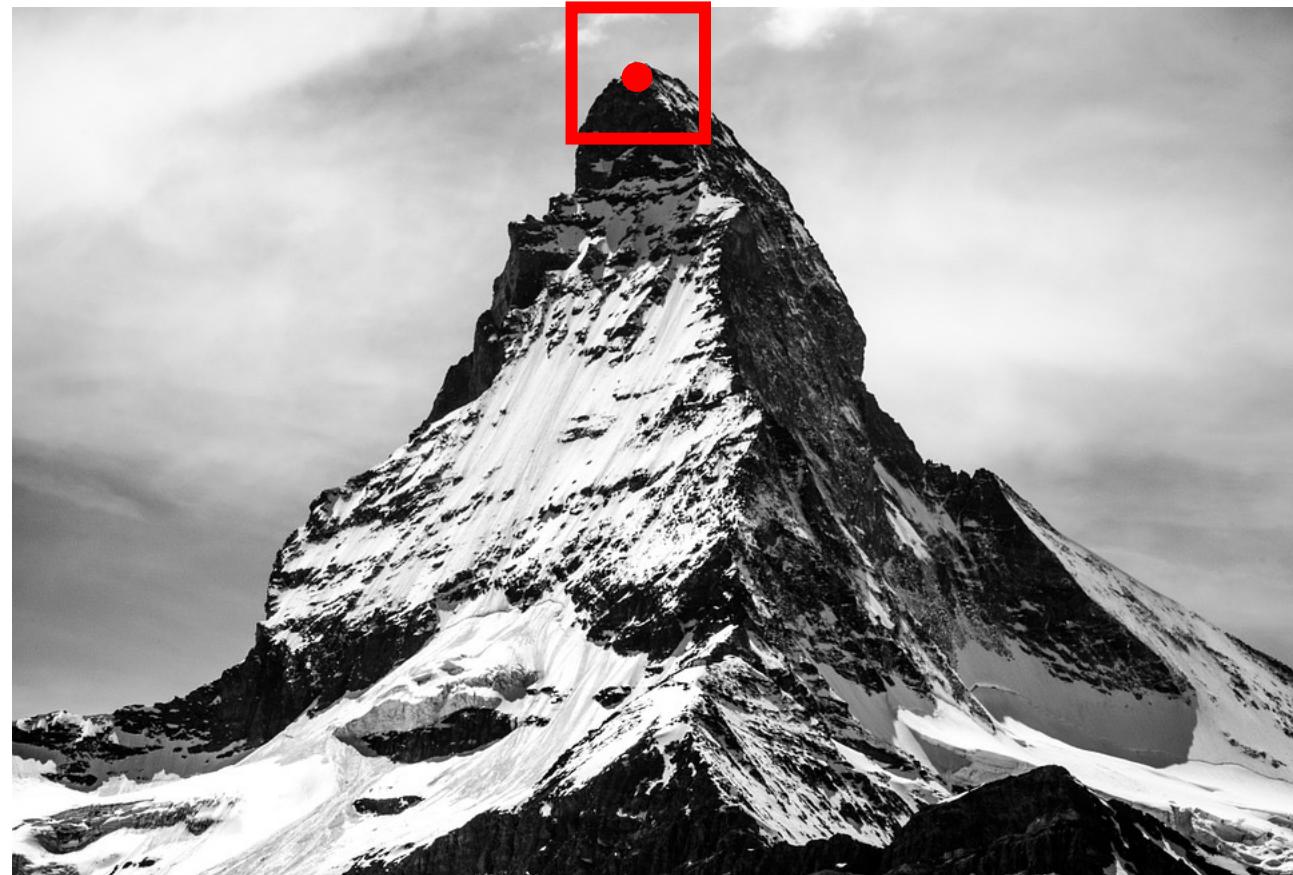
- **Rappel** : Pour la détection de coins, on a cherché la réponse maximale à plusieurs échelles.
- Conserver l'échelle de la réponse maximale :
Échelle caractéristique
- Utiliser des sous-régions dont la taille dépend de l'échelle caractéristique
- Ensuite, il faut **redimensionner** la sous-région à une taille fixe.



Descripteur par sous-régions orientées multi échelles

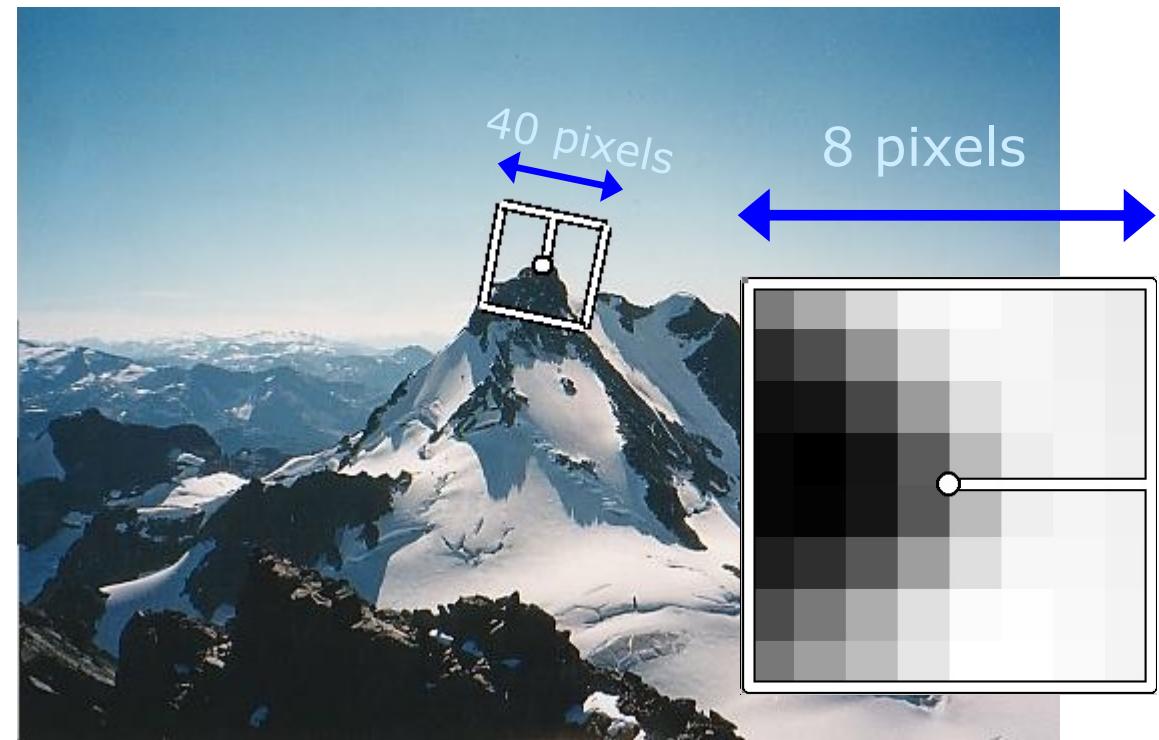
Multiscale Oriented PatcheS Descriptor (MOPS, 1)

- Décrire un coin par une sous-région autour du pixel localisé
- Invariance à l'échelle en utilisant l'échelle identifiée lors de la détection du coin
- Invariance à la rotation en utilisant l'orientation identifiée lors de la détection du coin
- Invariance photométrique en retirant la moyenne et en divisant par l'écart-type de l'intensité



Étapes pour calculer un descripteur MOPS

- Choisir une **fenêtre carrée** de taille 40x40 autour d'un point caractéristique détecté à l'**échelle caractéristique**
- Mettre à l'échelle 1/5 de sa taille (avec pré-filtrage)
- **Pivoter** la fenêtre horizontalement
- **Échantillonner** une fenêtre de taille 8x8 centrée sur le point caractéristique
- **Normaliser l'intensité** de la fenêtre en lui soustrayant sa moyenne, puis en divisant par l'écart-type dans la fenêtre



Rappel : Transformation d'images

- Que signifie « pivoter une sous-région » ?
- Chaque pixel a une coordonnée (x, y)
- Une rotation est représentée par une matrice R
- Nouvelles coordonnées des pixels

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = R \begin{bmatrix} x \\ y \end{bmatrix}$$

- $I'(x', y') = I(x, y)$

Quoi faire si la position transformée (x', y') est fractionnaire ?

- Inverser les calculs : Pour chaque pixel transformé, trouver le pixel original dans l'image.

$$\begin{bmatrix} x \\ y \end{bmatrix} = R^{-1} \begin{bmatrix} x' \\ y' \end{bmatrix}$$

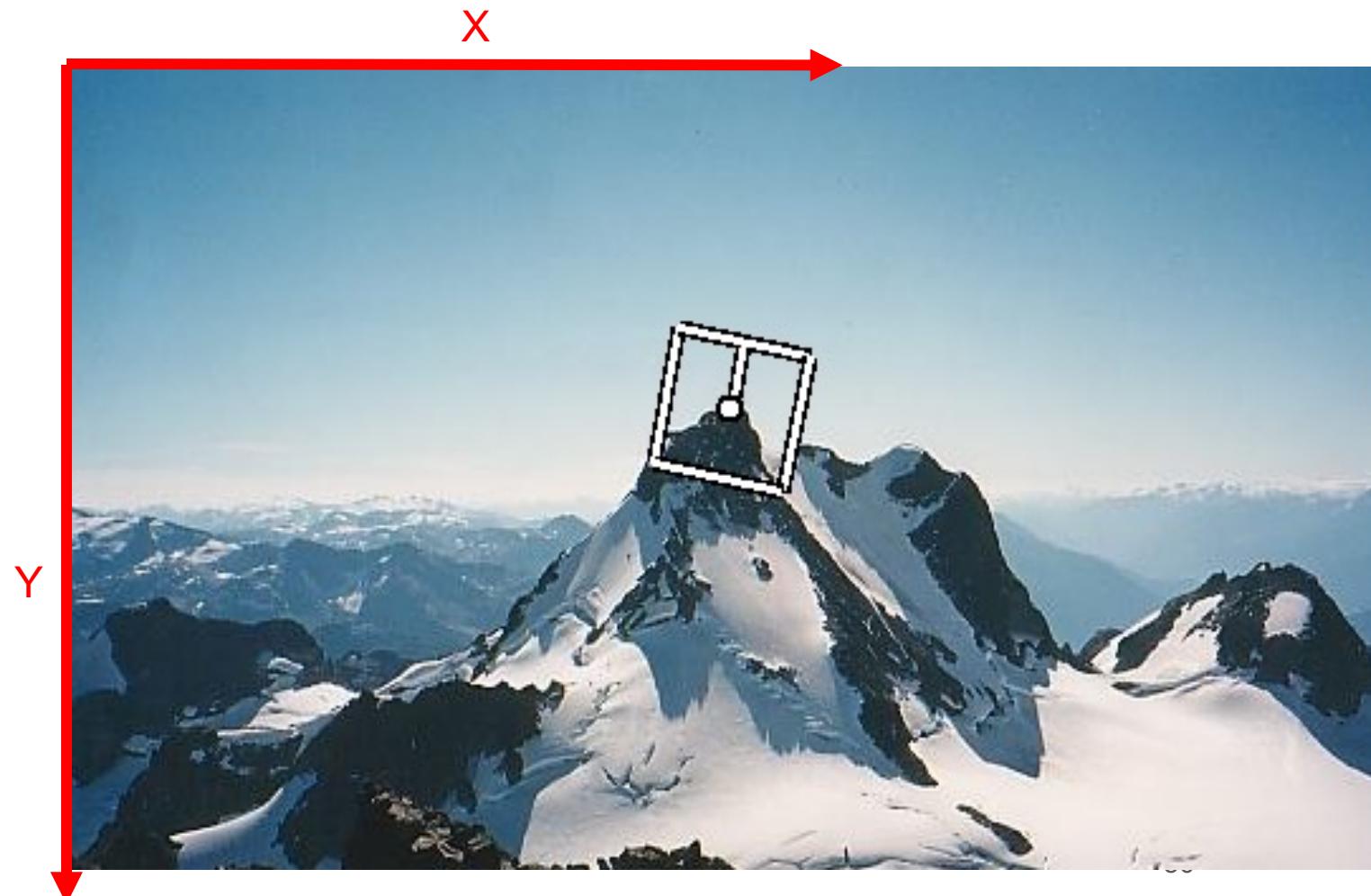
- Utiliser une interpolation pour trouver l'intensité)

Création des descripteurs MOPS

- Quelles transformations appliquer à l'image pour obtenir la transformation finale pour chaque MOPS?

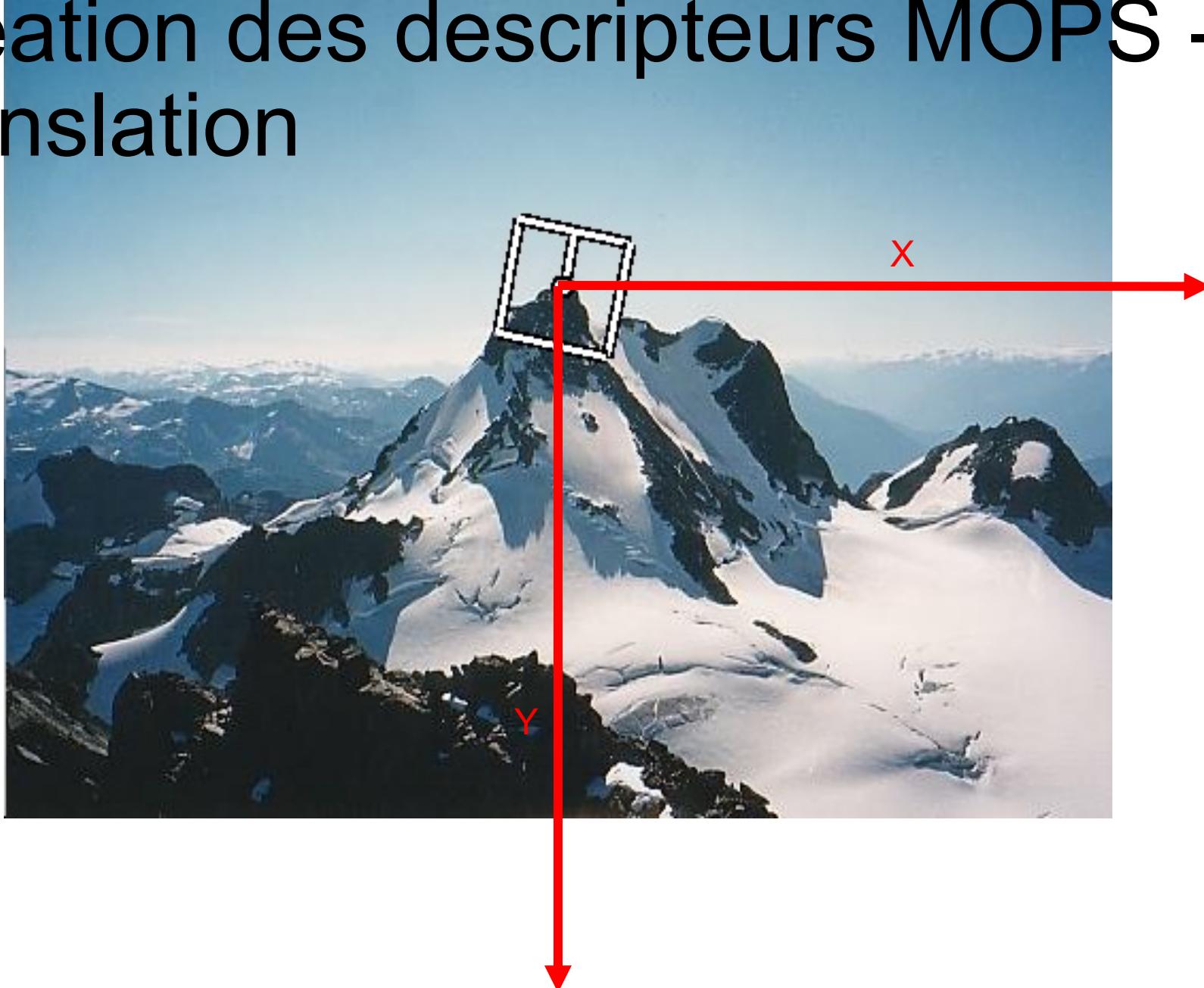
$$T = ?$$

- On peut combiner chaque transformation pour obtenir une seule opération à appliquer.

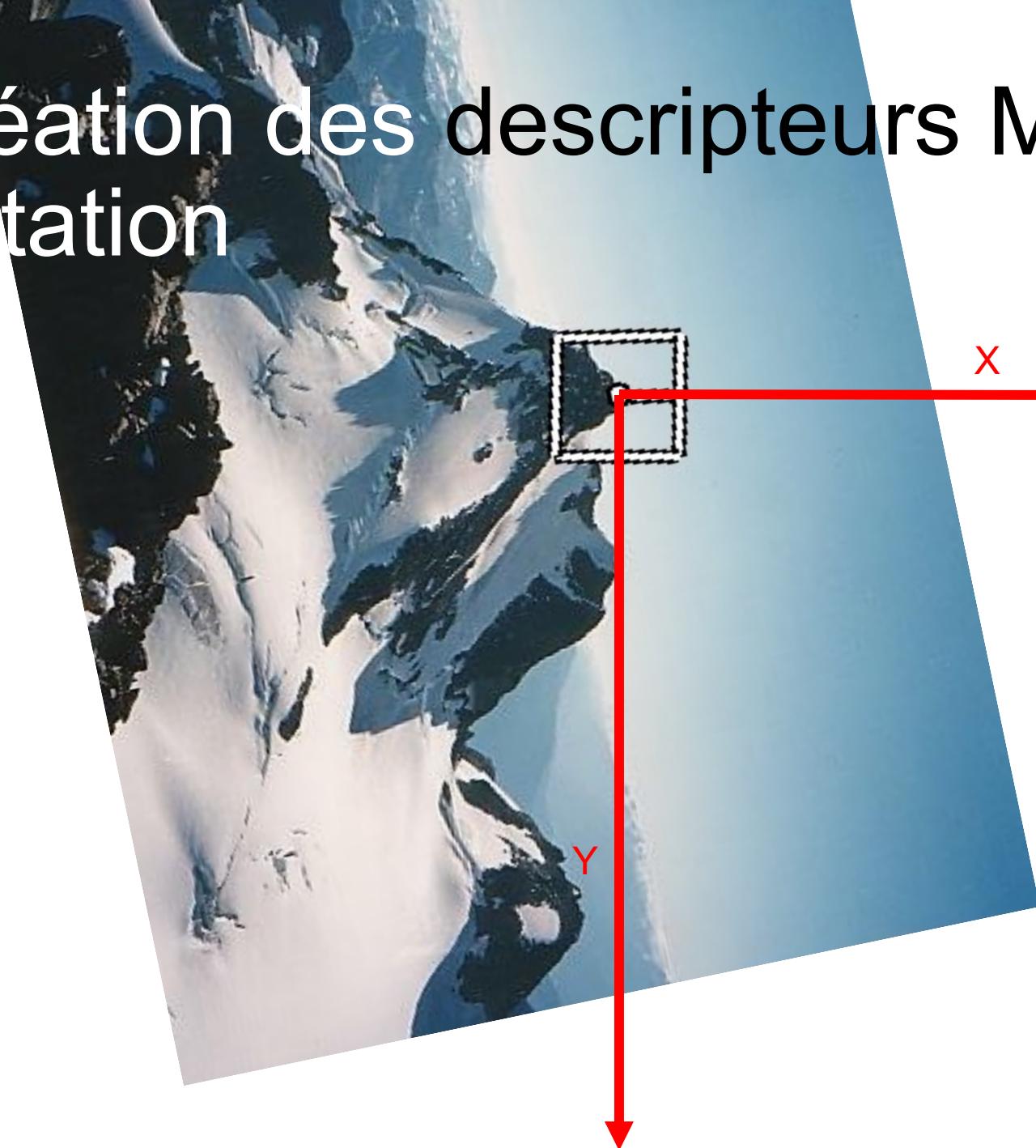


Création des descripteurs MOPS - Translation

$$T = M_{T1}$$

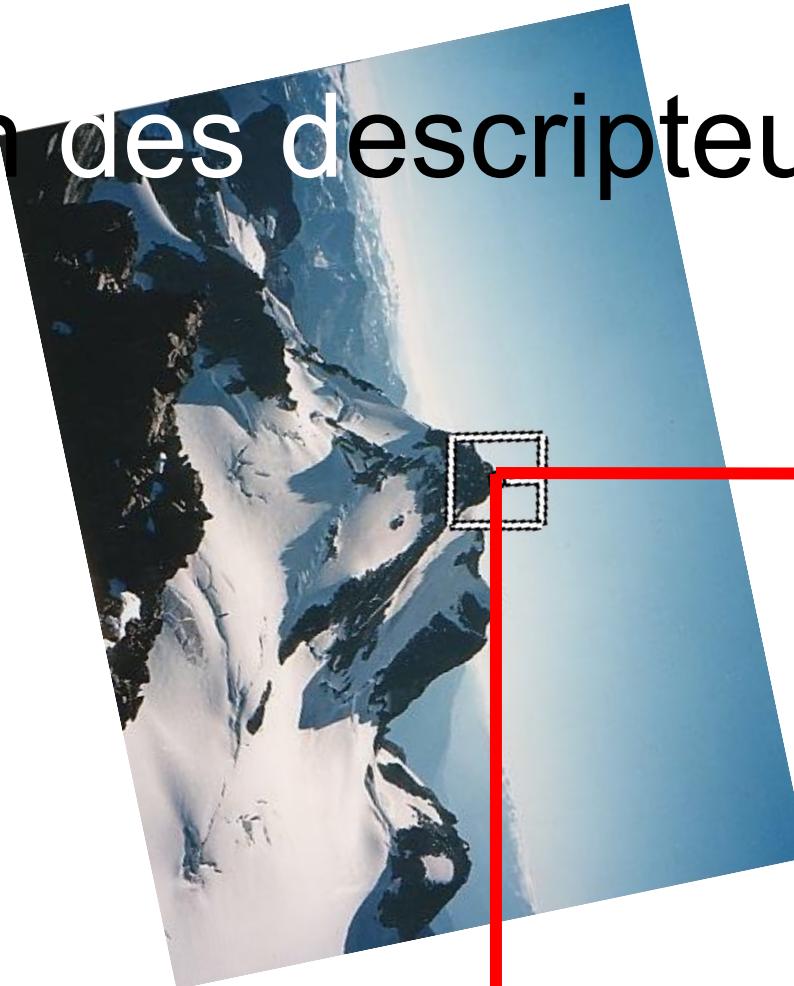


Création des descripteurs MOPS - Rotation



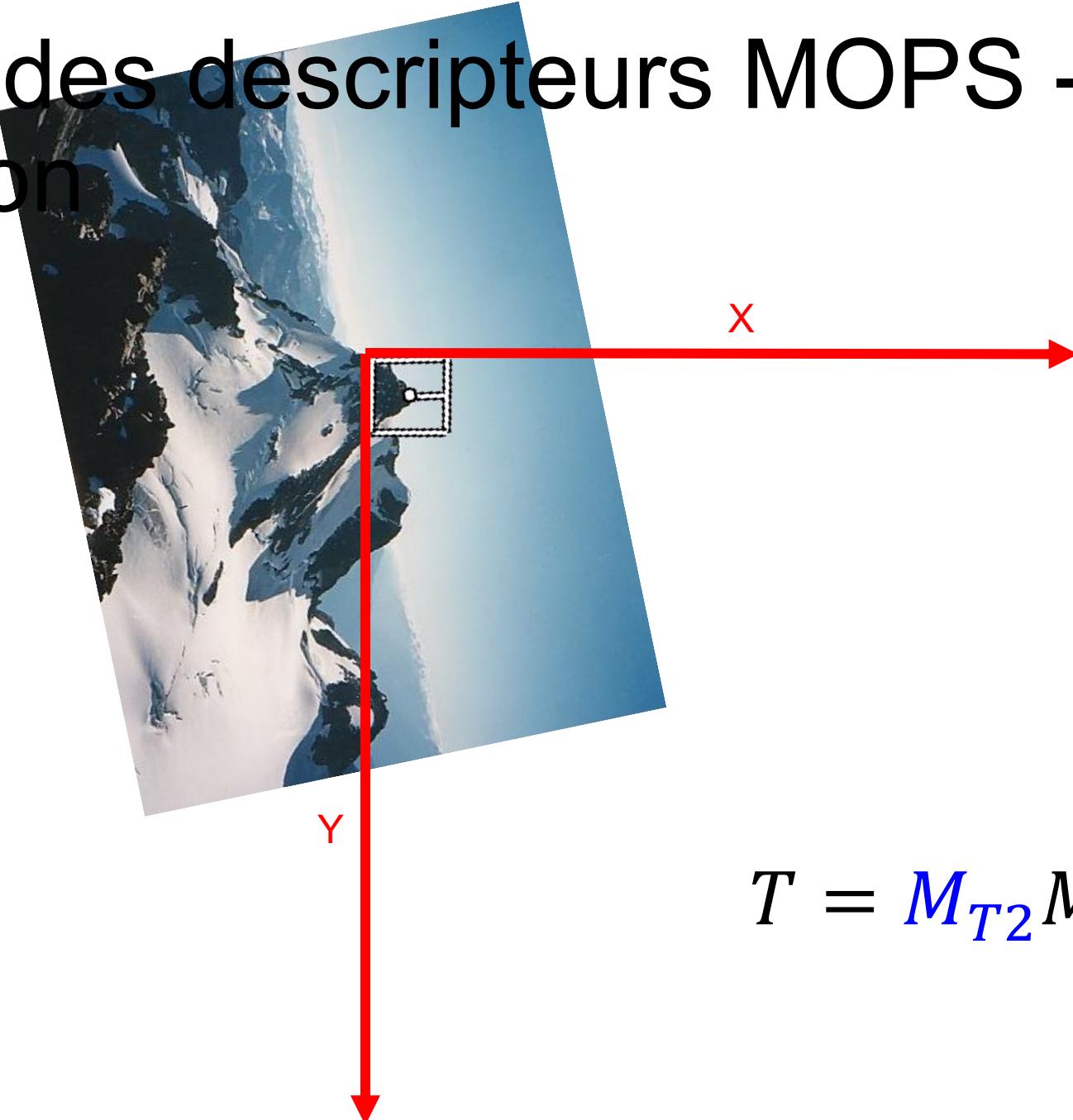
$$T = M_R M_{T1}$$

Création des descripteurs MOPS - Échelle



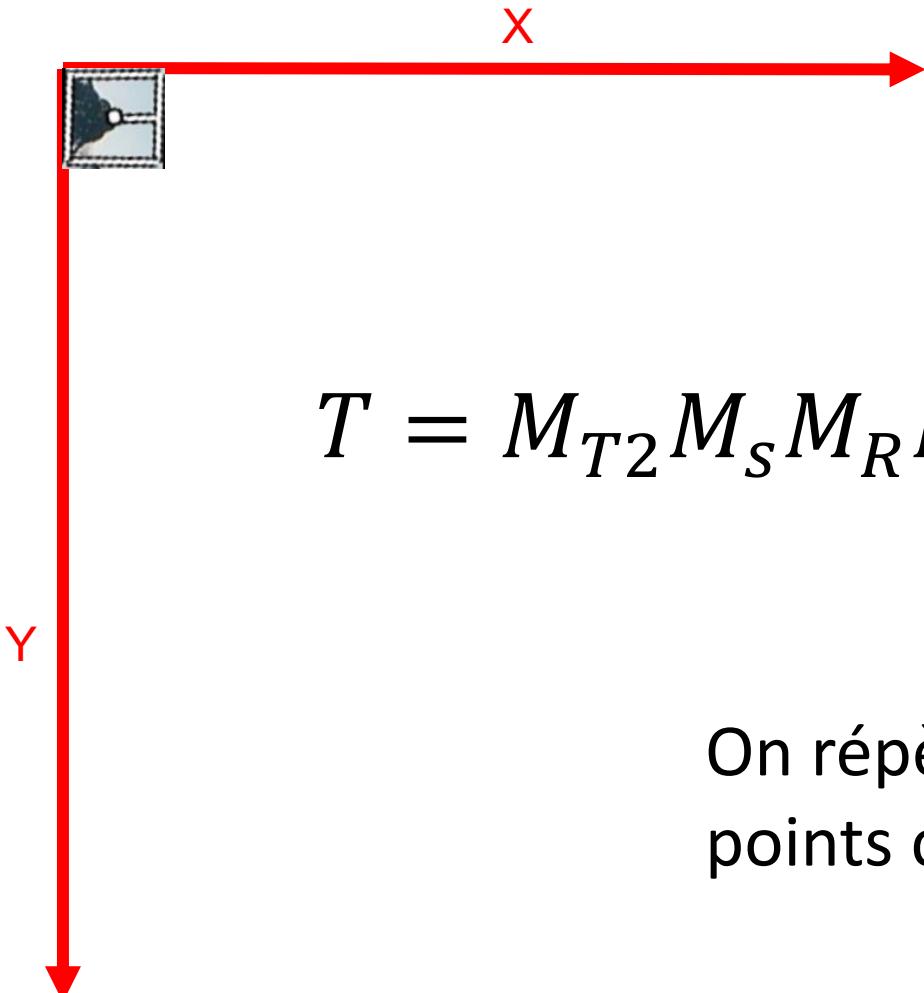
$$T = \mathcal{M}_S \mathcal{M}_R \mathcal{M}_{T1}$$

Création des descripteurs MOPS - Translation



$$T = M_{T2} M_S M_R M_{T1}$$

Création des descripteurs MOPS - Rognage



Détection à plusieurs échelles (MOPS)

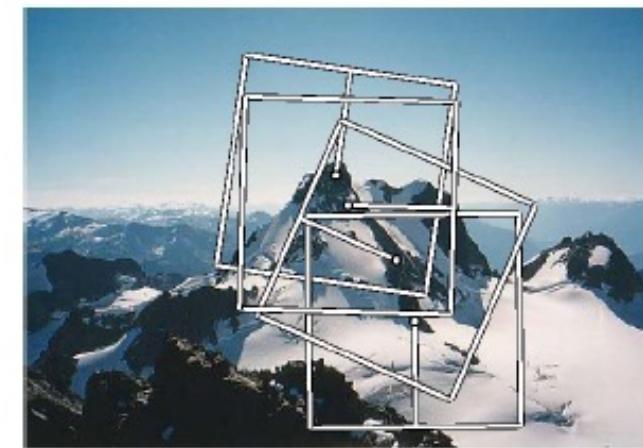
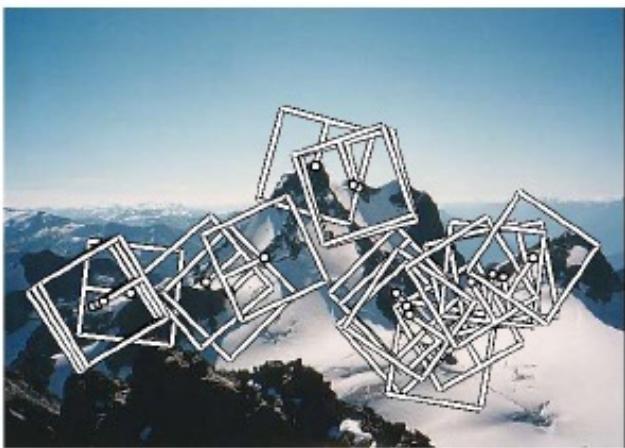


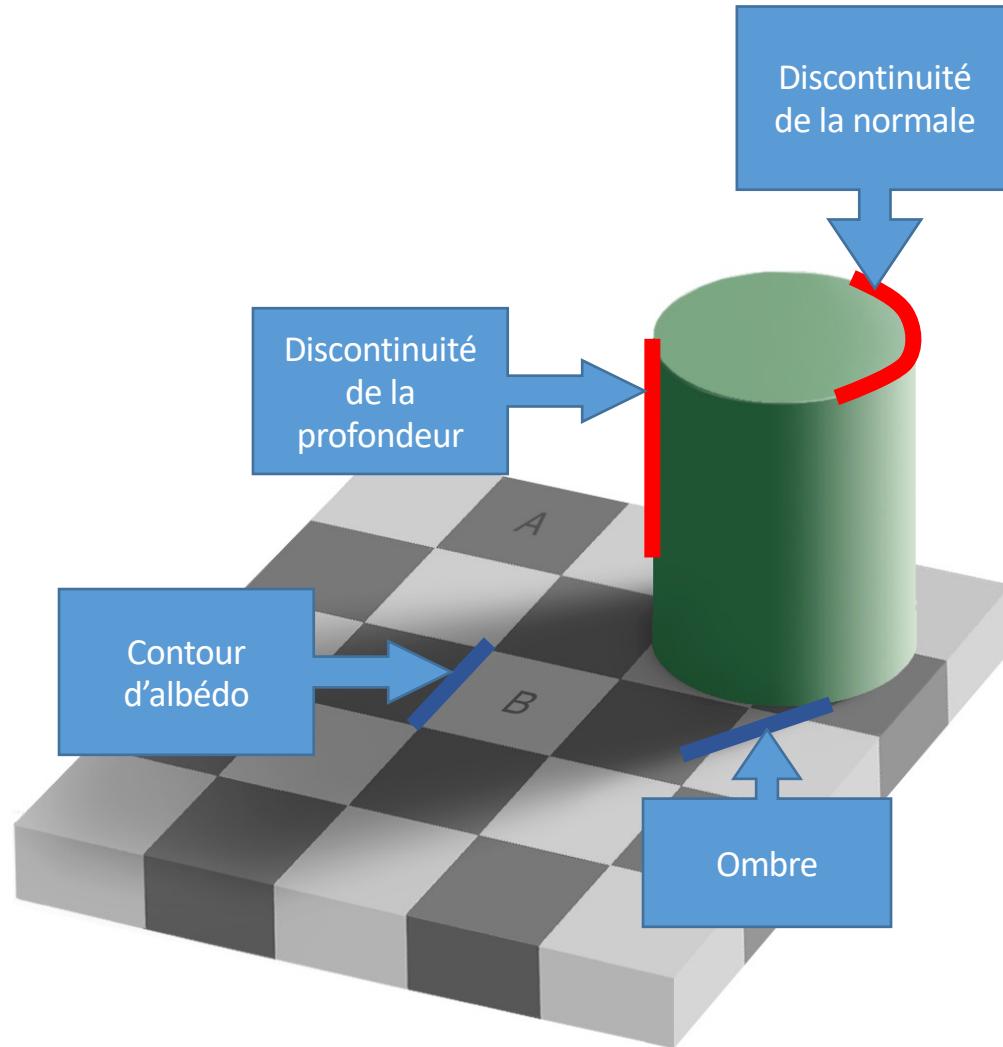
Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.

Couleur et illumination

- On a présenté un descripteur invariant face aux changements d'intensité additif et multiplicatif.
- Qu'en est-il pour les changements d'intensité plus complexes ?

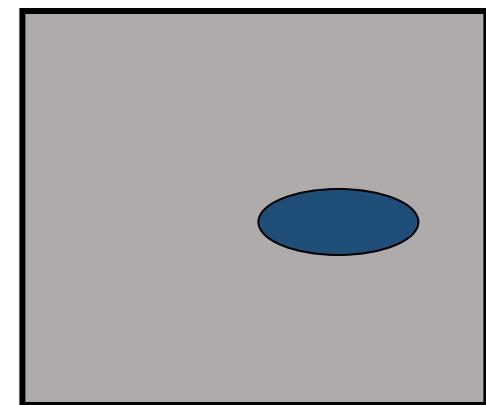
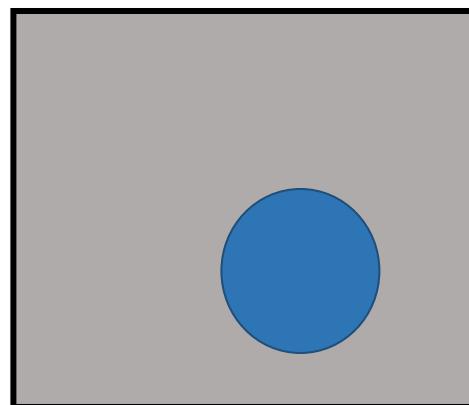
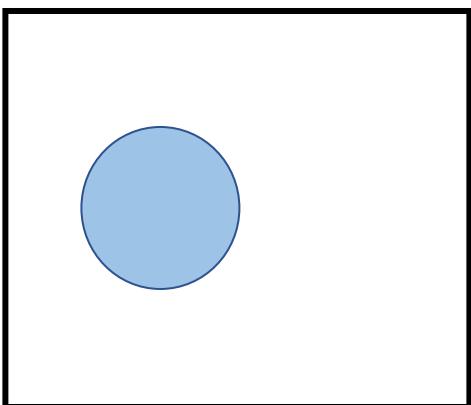


Meilleure représentation que la couleur : Contours



Rotation hors plan

- On a présenté un descripteur invariant face aux translations et rotations dans le plan de l'image
- Qu'en est-il pour les transformations géométriques plus complexes ?



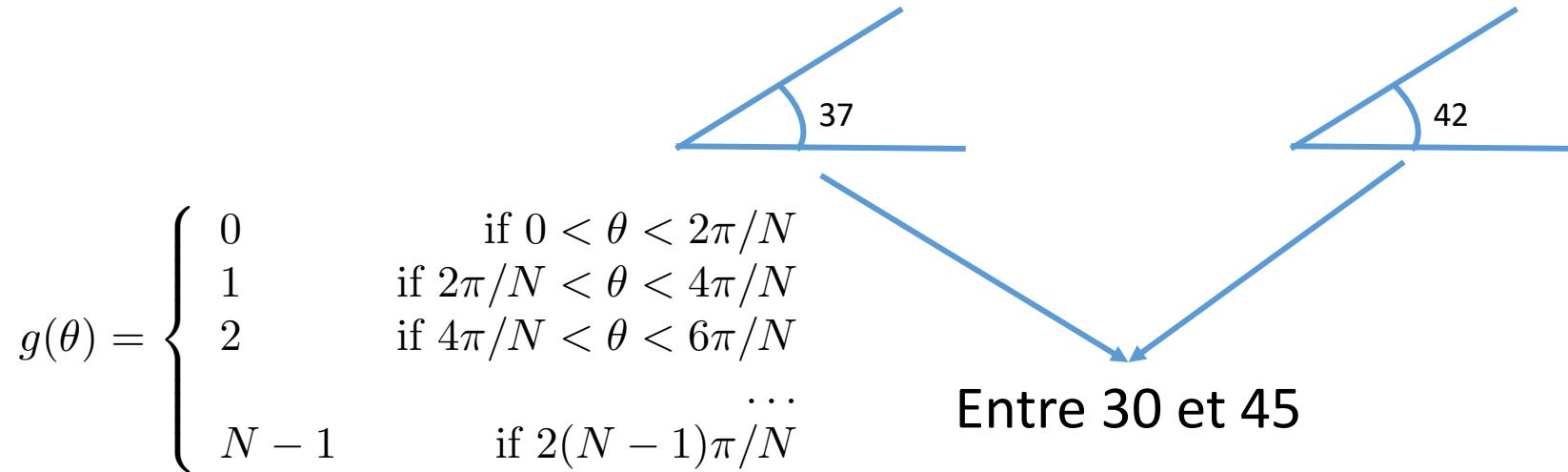
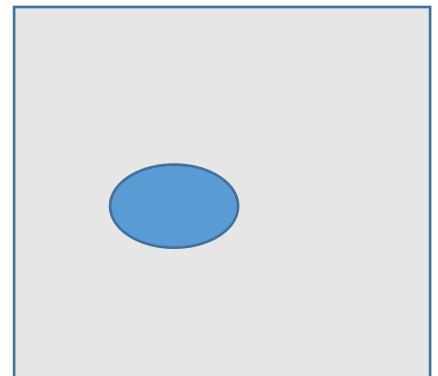
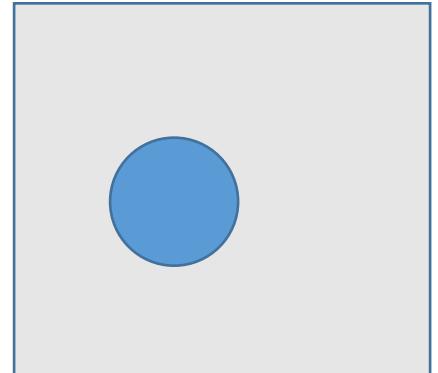
Rotation hors plan

Vers un meilleur descripteur des points caractéristiques

- Associer des ***motifs de contours***
 - Orientation des contours : indice sur la forme de l'objet
 - Invariant face à presque tous les types de transformations photométriques
- Résilience face à de ***petites déformations***
 - Les déformations peuvent déplacer les pixels, mais légèrement
 - Les déformations peuvent changer l'orientation des pixels, mais légèrement

Invariance face aux déformations

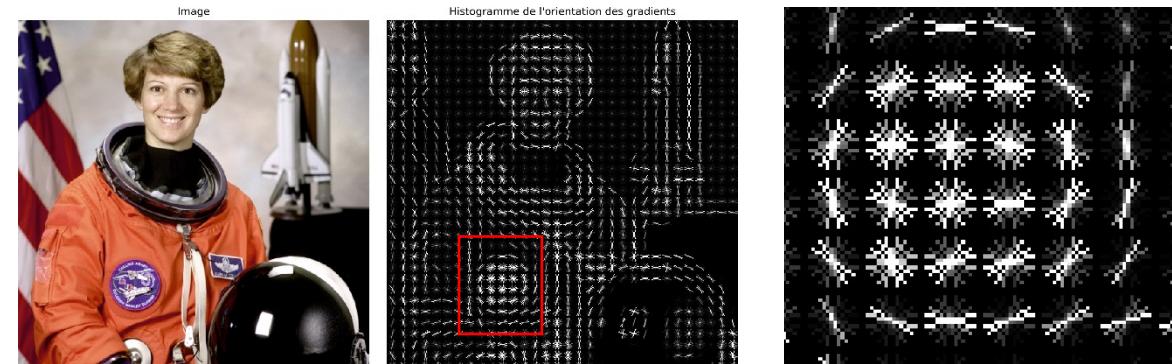
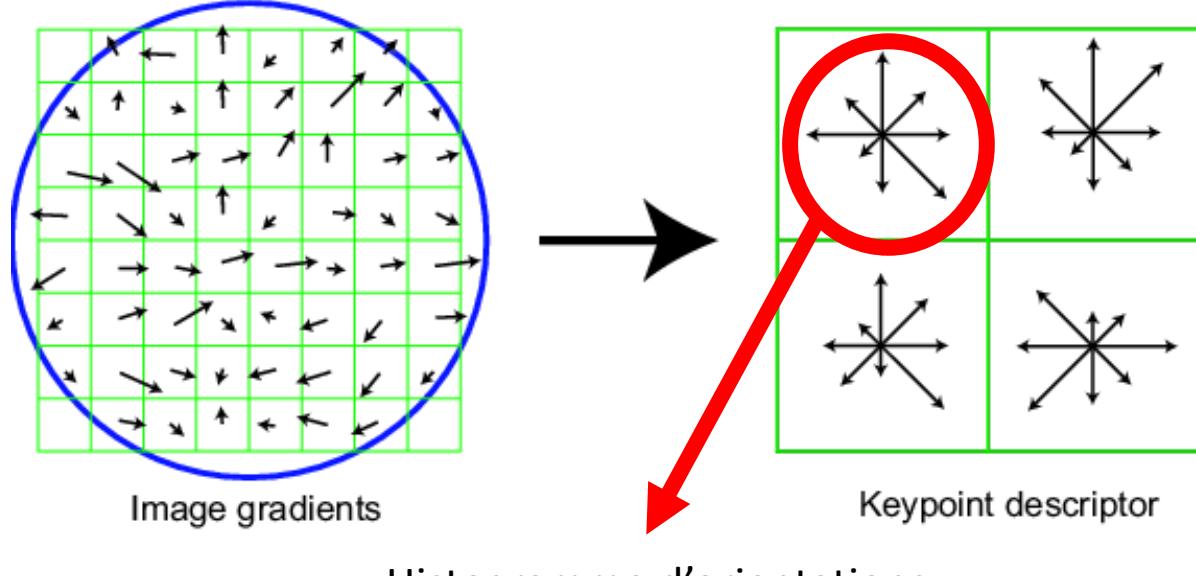
- L'orientation précise des contours n'est pas résiliente face aux rotations hors plan ou aux déformations.
- On peut toutefois **quantifier l'orientation**
- On ne considère qu'une approximation de l'orientation



Invariance face aux déformations

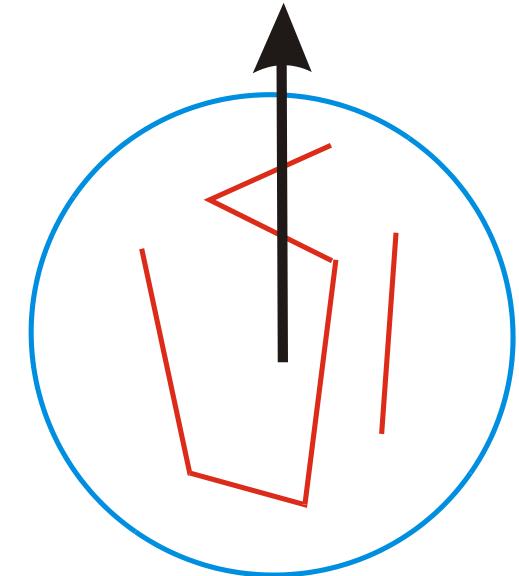
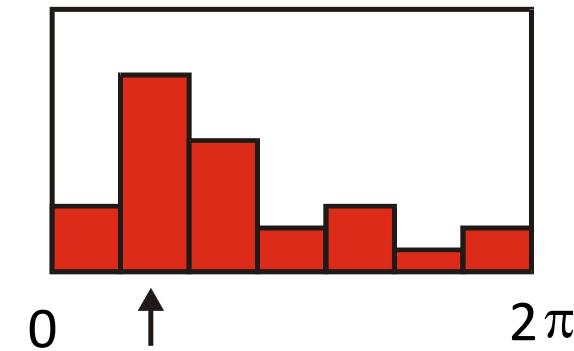
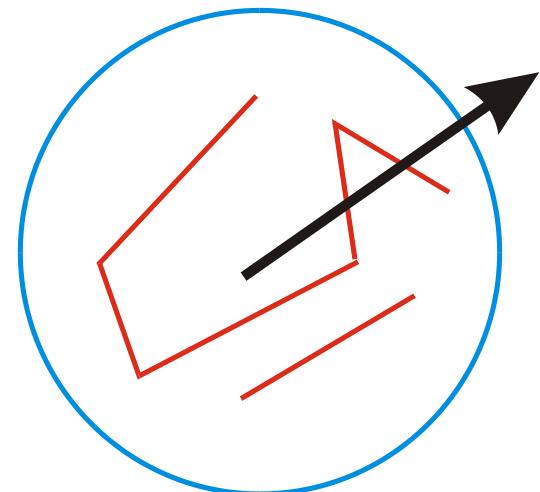
- Les **déformations locales** peuvent aussi légèrement déplacer la position des pixels
- Au lieu de retenir la position précise de chaque pixel, on peut conserver uniquement leur **position approximative**
- Division de la sous-région en grille de **cellules**
- Conserver la quantité de pixels pour chaque orientation et chaque cellule : **histogramme d'orientations**

Exemple skimage.feature.hog : [URL](#)

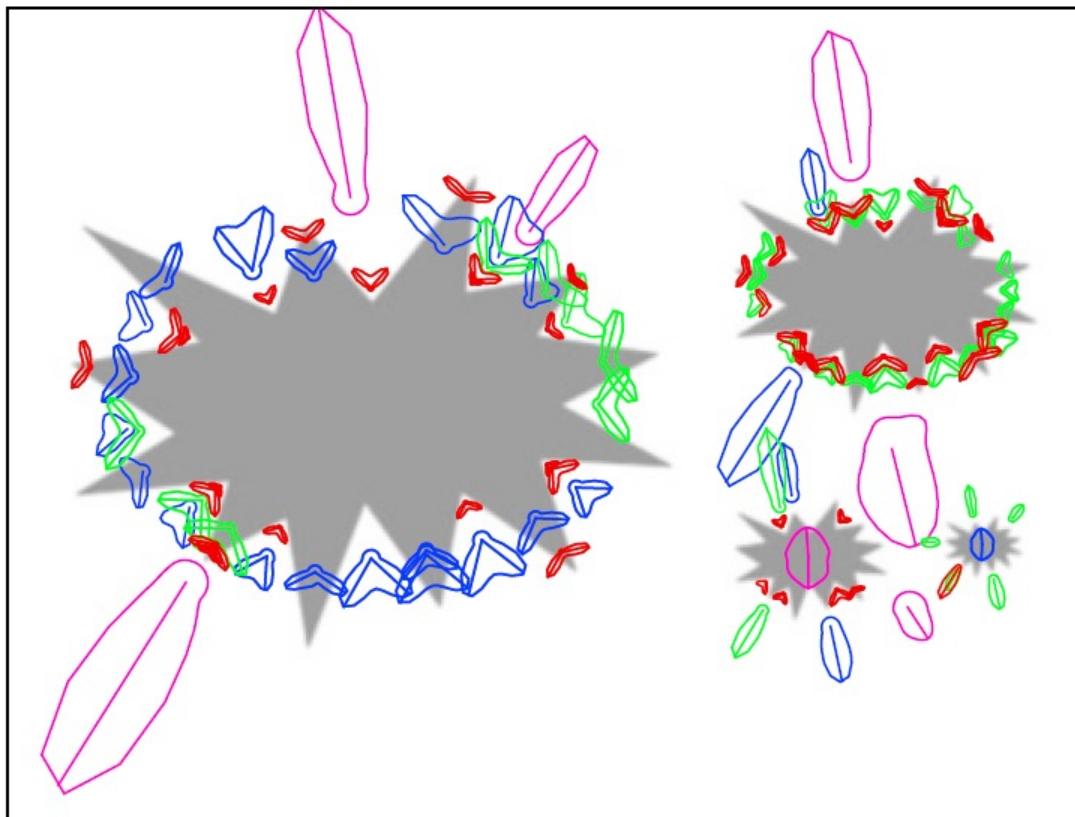


Invariance face aux rotations par normalisation de l'orientation

- Calcul de l'histogramme des orientations
- Sélectionner l'orientation dominante localement
 - Mode de l'histogramme
 - Vecteur propre de la matrice du second moment
- **Normalisation** : pivoter vers une orientation fixe



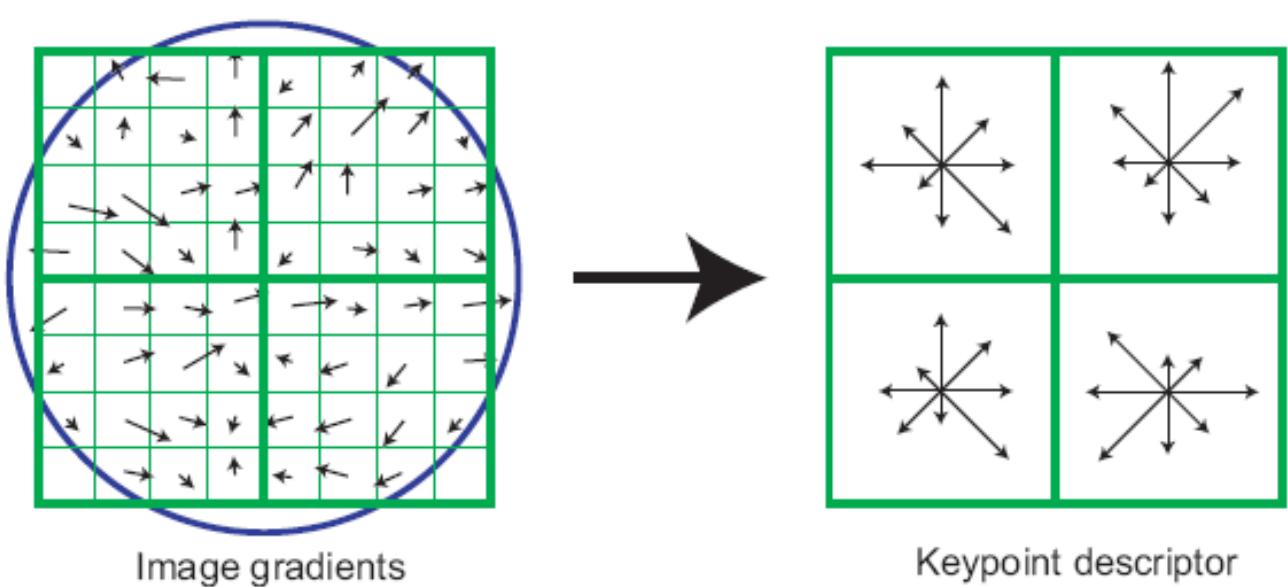
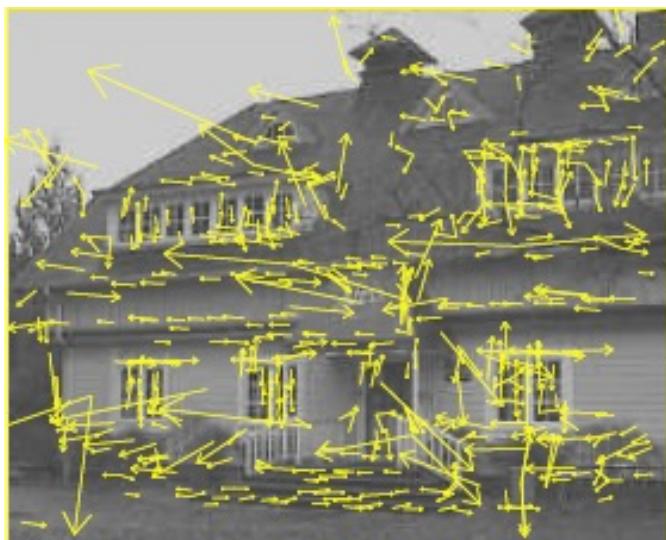
Exemple : Orientations dominantes



Source : BurgerVol3, Fig 7.21

Descripteur SIFT

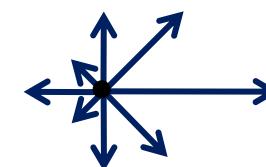
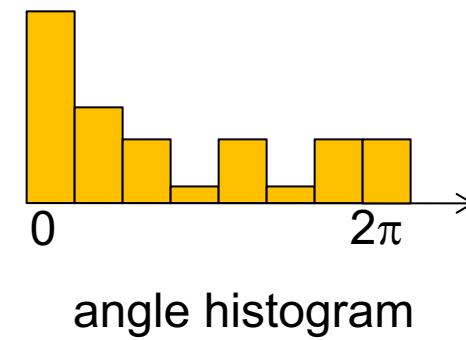
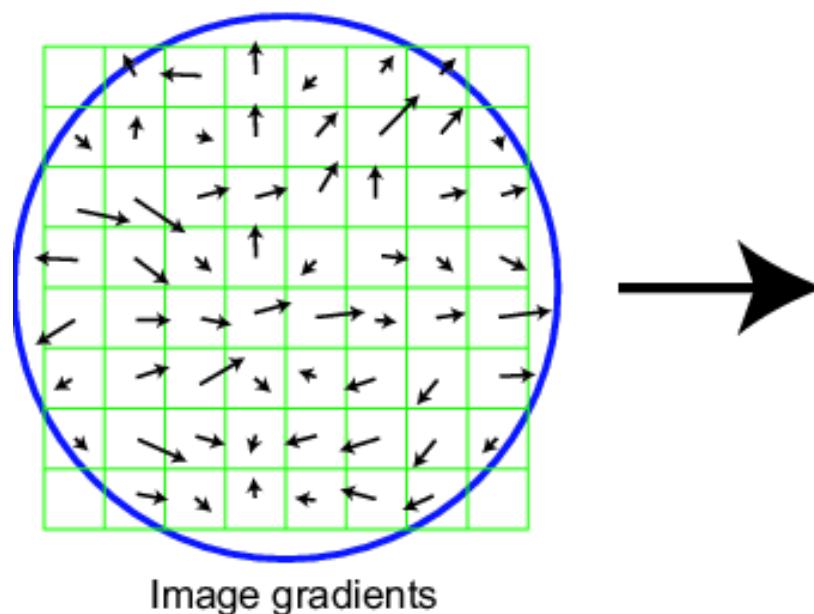
- ***Scale Invariant Feature Transform***
- Transformation de caractéristiques invariantes à l'échelle.
- SIFT est un détecteur et descripteur combiné



SIFT – Lowe IJCV 2004

SIFT – Idées de base

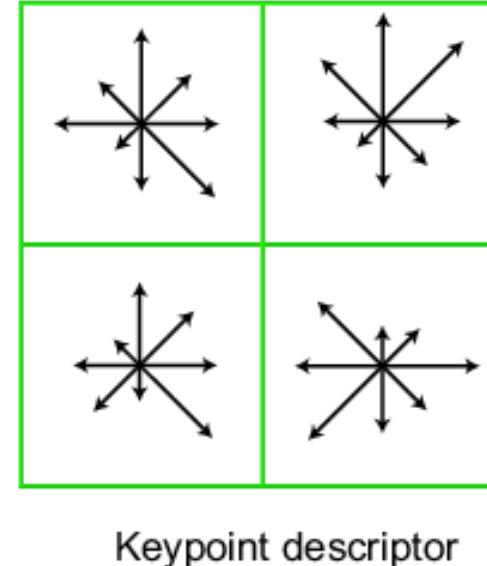
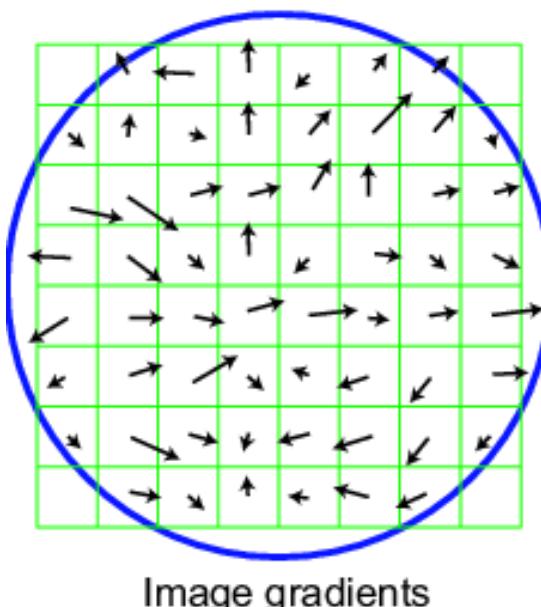
- **DoG** pour la **détection** spatiale/échelle des points caractéristiques 🐶
- Sélectionner une fenêtre carrée de taille 16x16 autour de la caractéristique
 - Calcul **l'orientation des contours** pour chaque pixel
 - Ignorer les contours faibles (**seuil** sur l'amplitude du gradient)
 - Créer un **histogramme des orientations** des contours restants



Source : N. Snavely

Descripteur SIFT

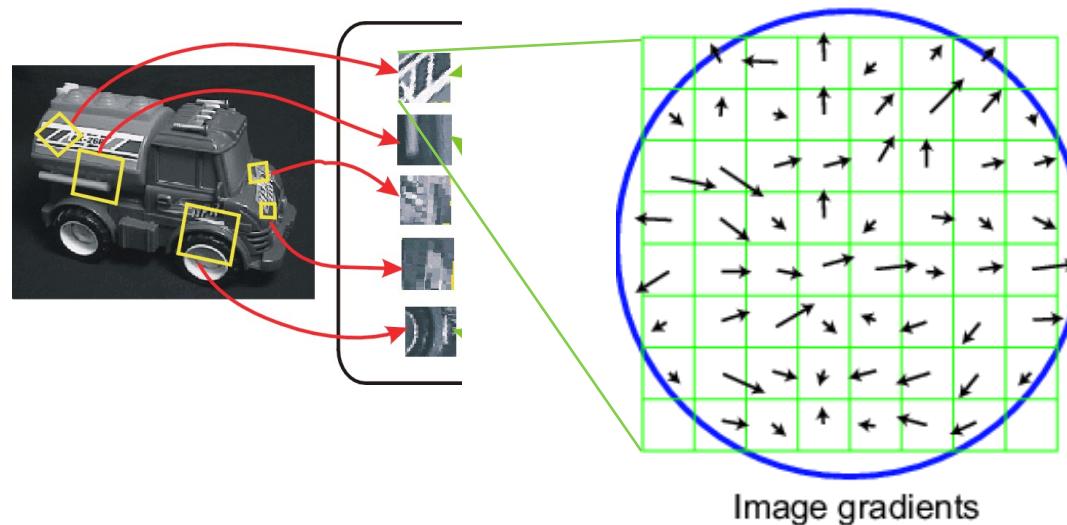
- Création de l'histogramme
 - Diviser la fenêtre 16x16 en une grille 4x4 de cellules
 - Calculer l'histogramme des orientations pour chaque cellule
 - On obtient un descripteur de taille 128 (16 cellules * 8 orientations)



Exemple 2x2
montré ici

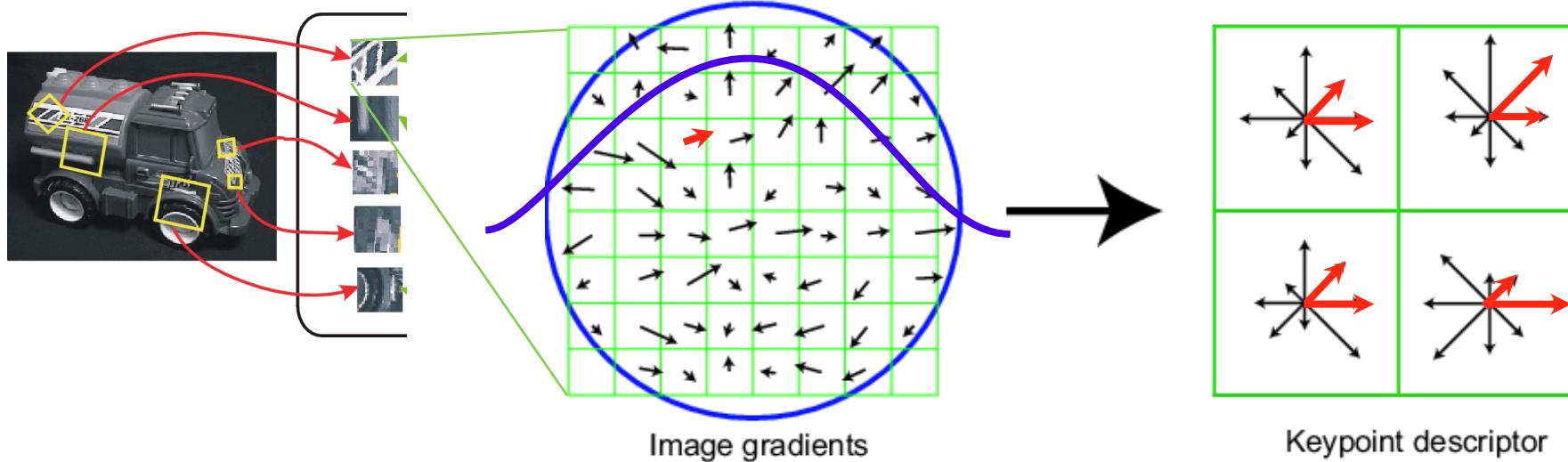
SIFT – Transformation vectorielle

- Calcul avec les fenêtres pivotées et mises à l'échelle en utilisant les orientations et échelles calculées pour chaque point caractéristique
- Ré-échantillonnage de la fenêtre
- Basé sur les gradients pondérés par une gaussienne



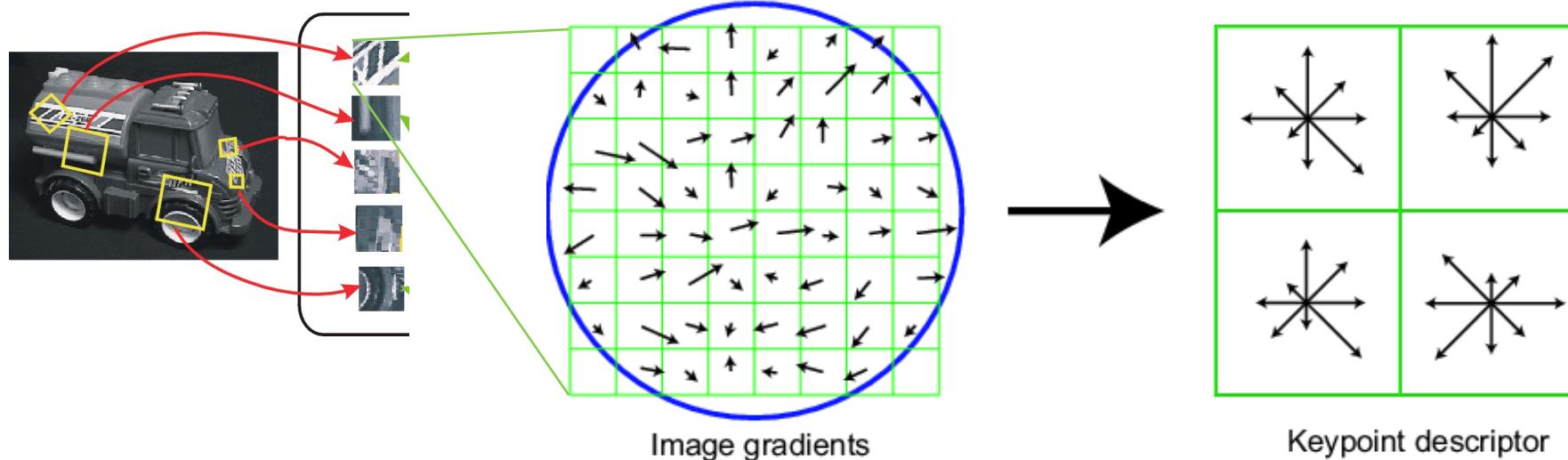
SIFT – Imposer des variations lentes

- Interpolation trilinéaire
 - Chaque gradient contribue à 8 valeurs d'histogramme
 - 4 spatiales x 2 orientations



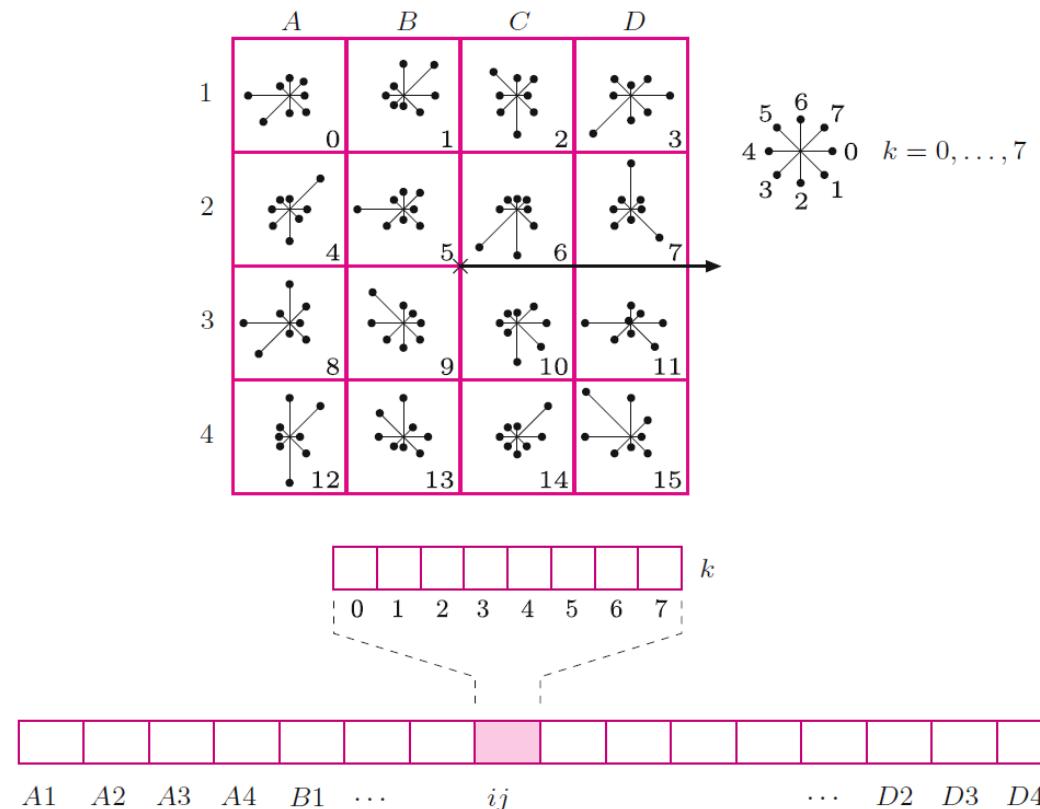
SIFT – Réduire l'effet de l'illumination

- Vecteur de dimension 128 normalisé à 1
- Seuillage des amplitudes des gradients pour éviter que les grands gradients aient une influence excessive
 - Après la normalisation, borner les gradients > 0.2
 - Renormaliser



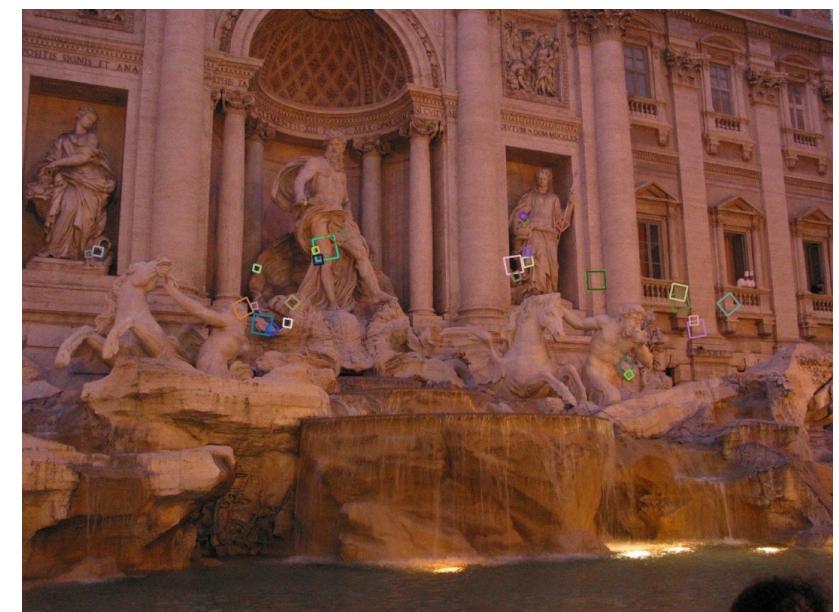
SIFT – Structure du descripteur

- Pour $n_{spat} = 4$ et $n_{angl} = 8$, on forme un descripteur vectoriel de taille $4 \times 4 \times 8 = 128$ pour chaque point caractéristique
- C'est le **descripteur SIFT**



Propriétés de SIFT

- Une technique de paireage extraordinairement **robuste**
- Peut considérer des **changements de points de vue** (jusqu'à une rotation de 60 degrés hors plan)
- Peut considérer des **changements d'illumination** significatifs
- Rapide et efficace, peut même être en **temps réel**
- Plusieurs implémentations disponibles



Source : Steve Seitz

Résumé de l'algorithme SIFT

- Construction de l'espace échelle
- Détection des points caractéristiques initiaux
- Amélioration de la précision des points caractéristiques
- Suppression des points non désirables
- Calcul de l'orientation dominante des points caractéristiques
- Calcul d'un descripteur par point caractéristique

Autres méthodes

- Scikit-image
 - Sous module : feature
 - Canny
 - Daisy
 - HOG *Histo. générée orienté*
 - ...
- Autres méthodes pour étudier la texture, les formes binaires, etc
- [Documentation en ligne](#)
- OpenCV : Librairie de vision par ordinateur (C++ et Python)
 - DéTECTEUR de Harris
 - DéTECTEUR de coins de Shi-Tomasi
 - SIFT
 - SURF (Version plus rapide)
 - FAST (Version temps réel pour SLAM)
 - BRIEF (+ rapide, - mémoire)
 - ORB (version gratuite sans brevet)
- [Documentation en ligne](#)

Méthodes récentes : CNNs

- Les méthodes récentes (depuis 2016) utilisent des **réseaux de neurones à convolution** pour détecter les caractéristiques.
- **LIFT**, *learned invariant feature transforms* ([Yi, 2016](#)), **SuperPoint**, *self-supervised interest point detection and description* ([DeTone, 2018](#)), et **LF-Net**, *learning local features from images* ([Ono, 2018](#)), optimisent conjointement les détecteurs et les descripteurs dans un pipeline unique (multi-têtes)
- **AffNet** ([Mishkin, 2018](#)), qui détecte les régions covariantes affines appariables;
- **Key.Net** ([Barroso-Laguna, 2019](#)), utilise une combinaison de caractéristiques classiques et apprises avec un CNN
- **D2-Net** ([Dusmanu, 2019](#)), **R2D2** ([Revaud, 2019](#)), et **D2D** ([Tian, 2020](#)), extraient des descripteurs locaux denses et conservent ceux qui ont une grande saillance ou répétabilité.

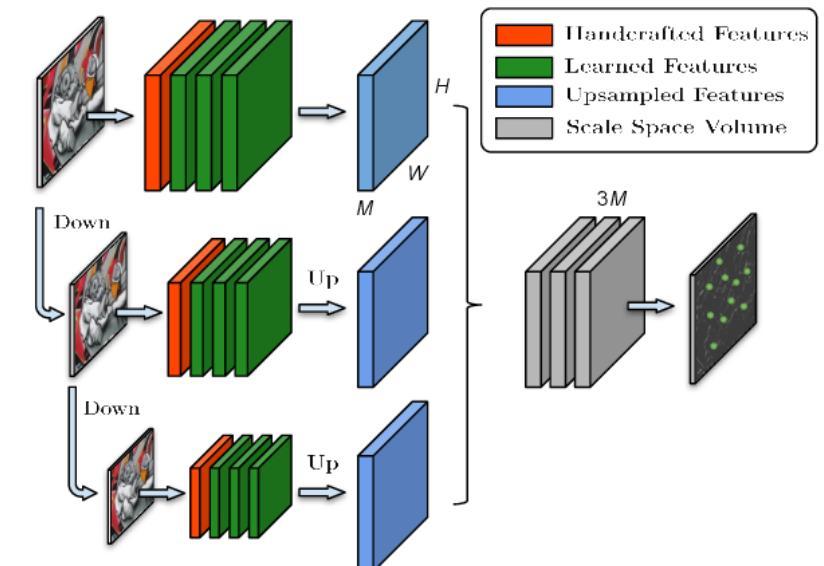
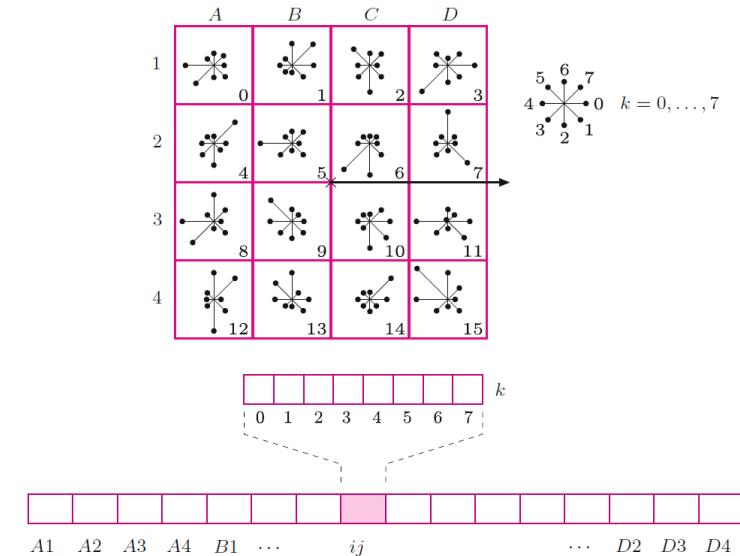


Figure 1: The proposed Key Net architecture combining learned and hand-crafted features. The figure shows three parallel processing paths, each starting with a small input image and ending with a feature map. The paths are as follows:

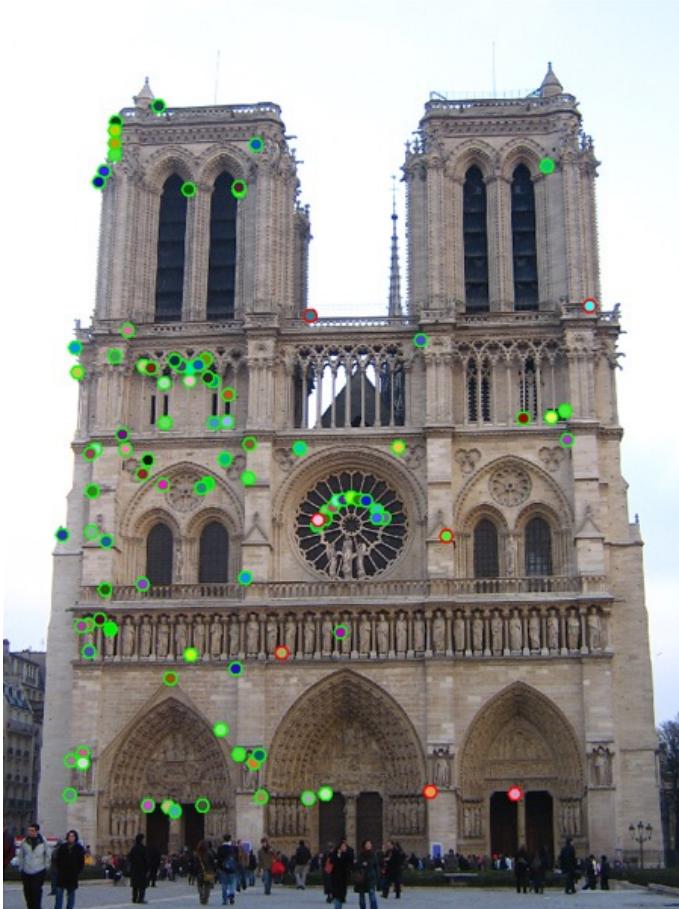
Key.Net ([Barroso-Laguna, 2019](#))

Résumé

- Détection de points caractéristiques : répétable et distinct
 - Coins, taches, régions stables
 - Harris, LoG, DoG
- Descripteurs : Robuste et sélectif
 - Histogramme spatial des orientations
 - SIFT et ses variantes sont généralement bonnes pour la reconstruction de mosaïques et la reconnaissance d'image
 - Autres méthodes possibles, à expérimenter



Quels points caractéristiques correspondent ?

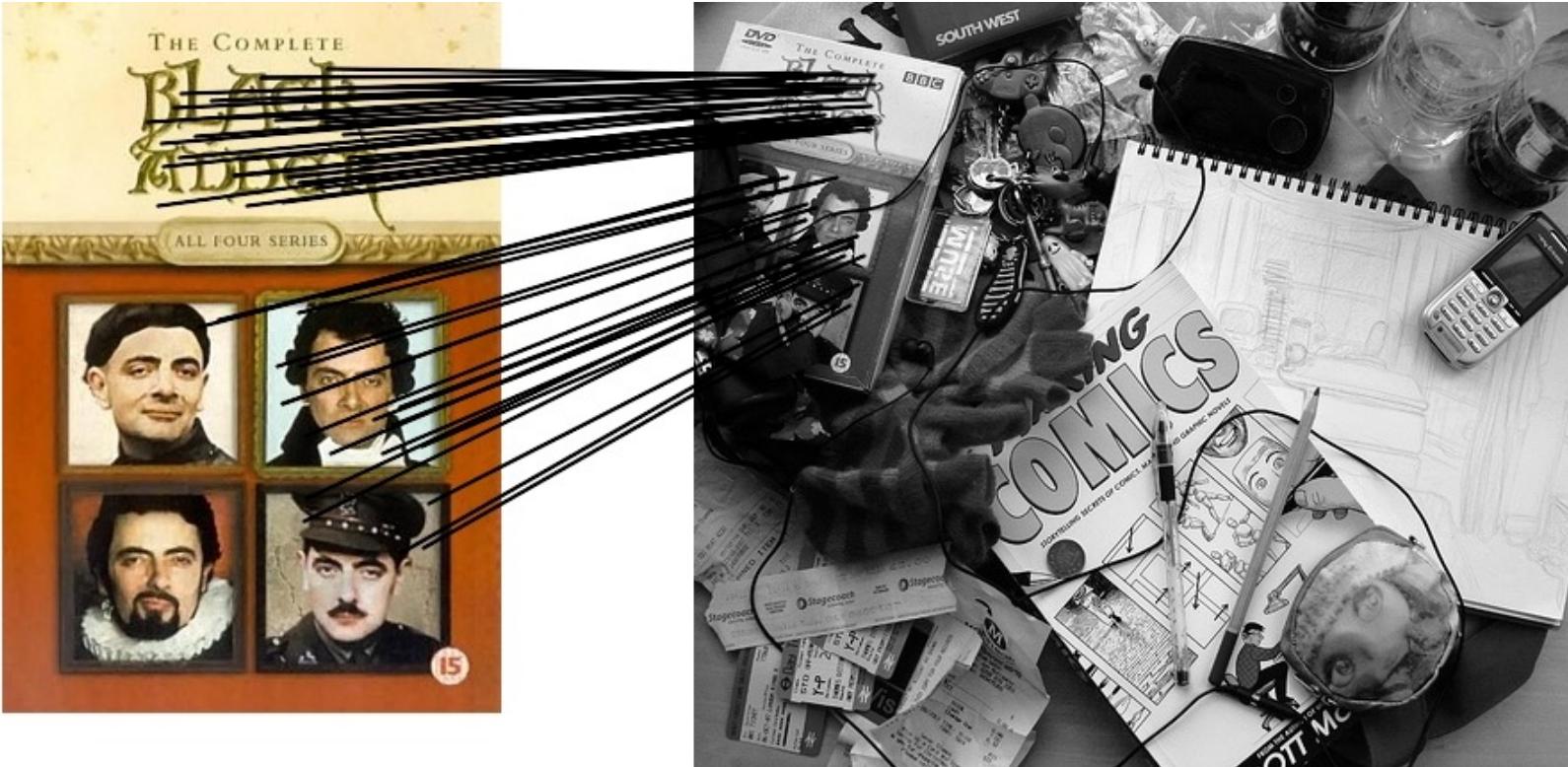


Source : N. Snavely

Correspondance de caractéristiques (*Feature matching*)

- Étant donné un point caractéristique dans I_1 , comment trouver la meilleure correspondance dans I_2 ?
 1. Définir une fonction de distance qui compare deux descripteurs (SSD, NCC, ...)
 2. Tester toutes les caractéristiques dans I_2 , et trouver celle avec la distance minimale

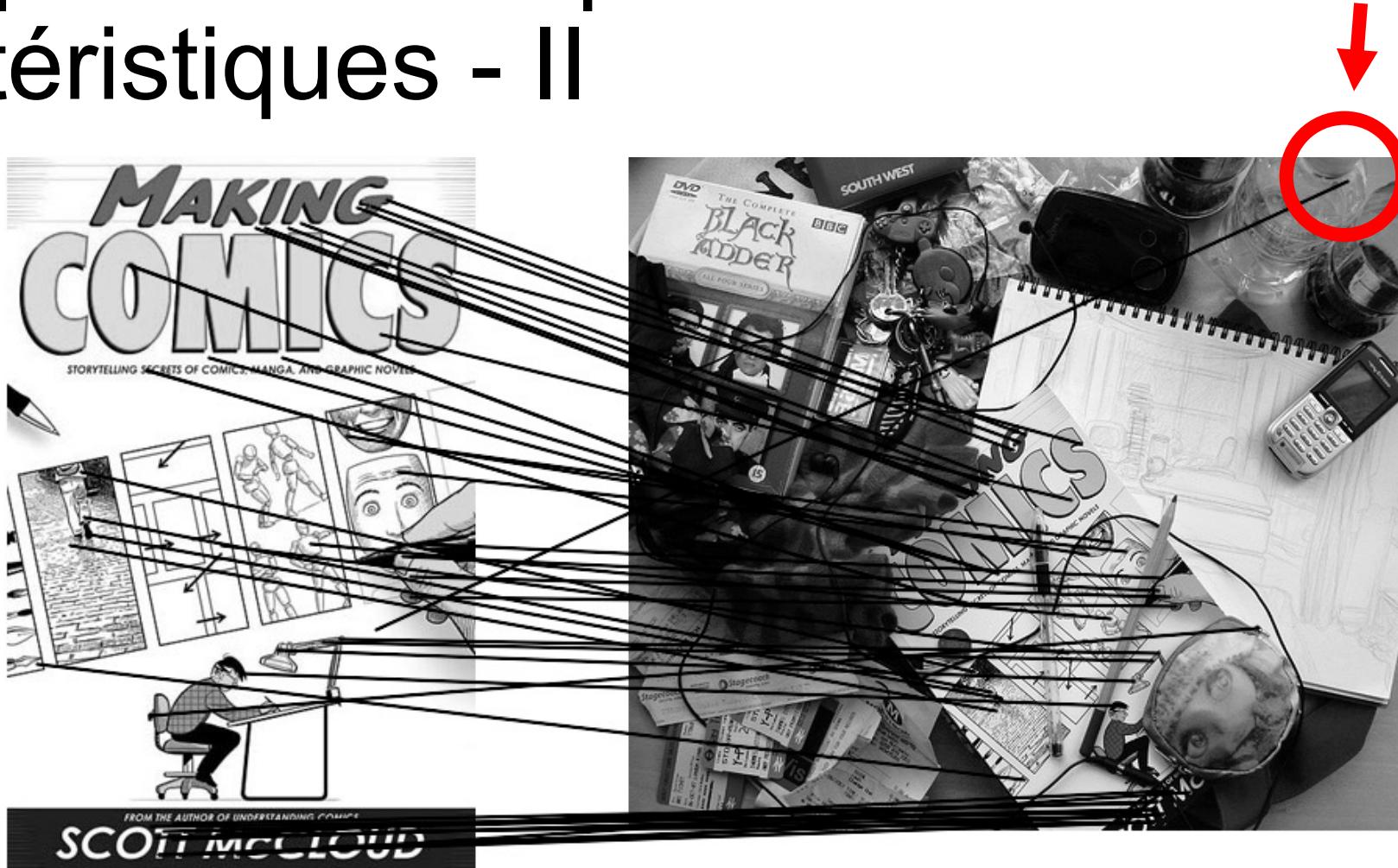
Exemple de correspondance entre caractéristiques - I



58 correspondances (seuillage par le ratio)

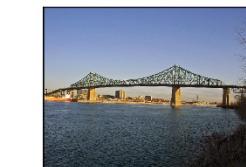
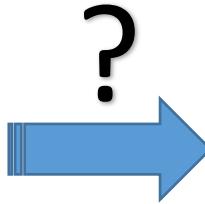
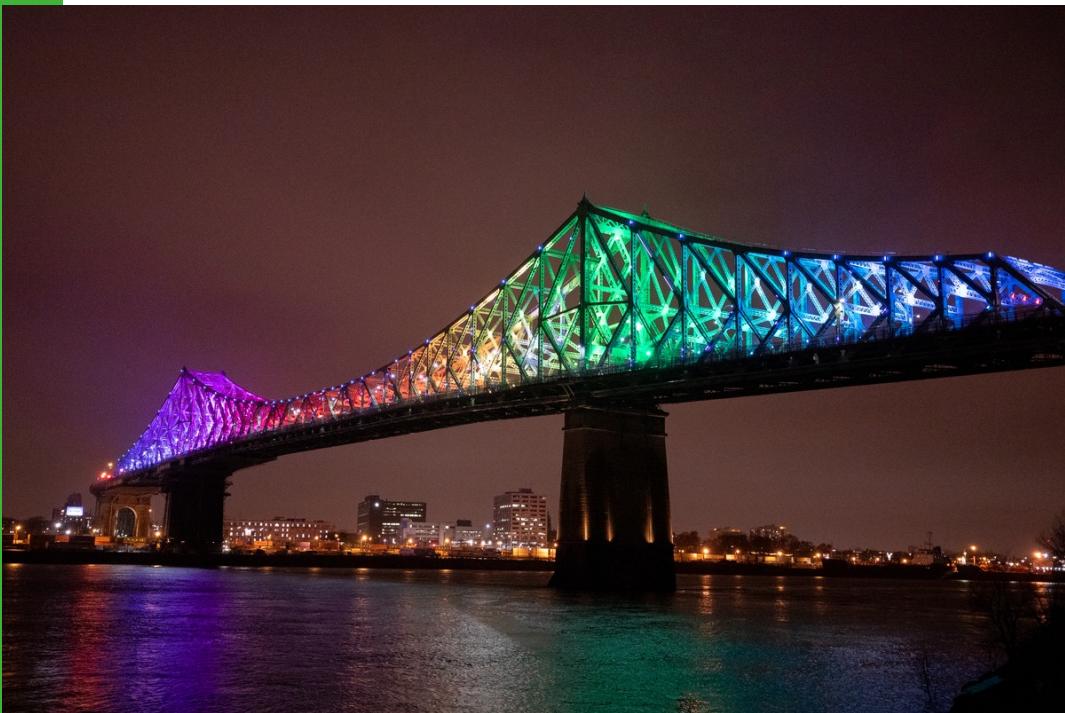
Détection
aberrante

Exemple de correspondance entre caractéristiques - II



51 correspondances (seuillage par le ratio)

Exemple : Identification du pont mystère (1)



Exemple : Identification du pont mystère (2)

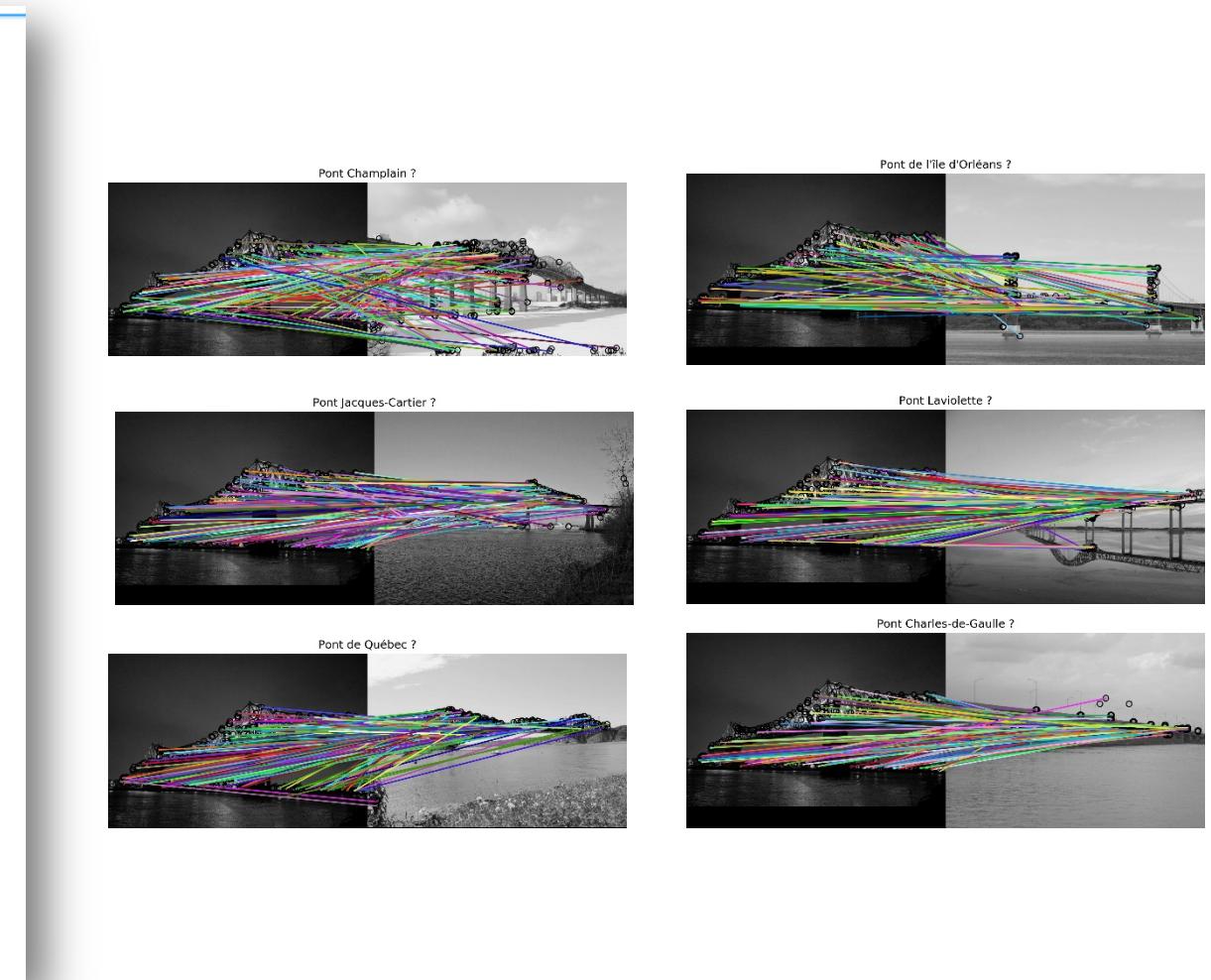
```
from skimage.feature import (match_descriptors, ORB, plot_matches)
from skimage.color import rgb2gray

img_mystere_gray = rgb2gray(img_mystere)
descriptor_extractor = ORB()

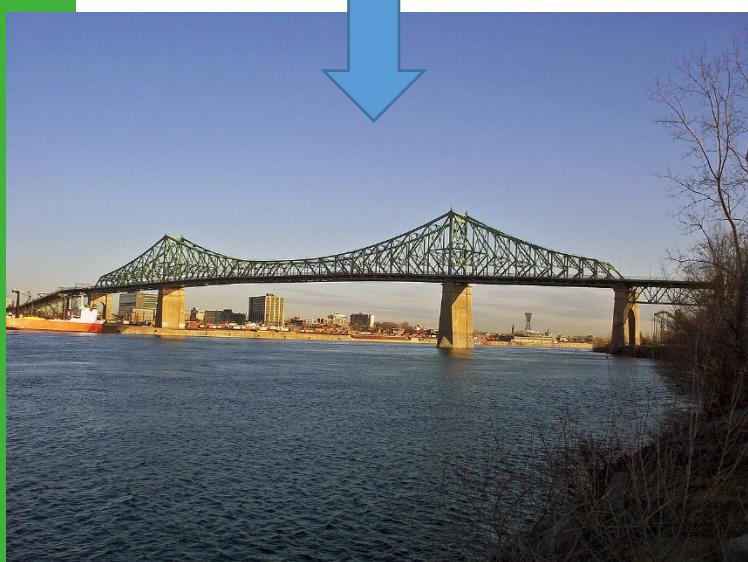
# Extraction des descripteurs et points caractéristiques
descriptor_extractor.detect_and_extract(img_mystere_gray)
keypoints_myst = descriptor_extractor.keypoints
descriptors_myst = descriptor_extractor.descriptors

# Process each image in database
n_matches = []
for i, img in enumerate(img_list):
    img = rgb2gray(img)
    descriptor_extractor.detect_and_extract(img)
    keypoints_img = descriptor_extractor.keypoints
    descriptors_img = descriptor_extractor.descriptors

    matches = match_descriptors(descriptors_myst, descriptors_img)
    n_matches.append(len(matches))
```



Exemple : Identification du pont mystère (3)



Pont	# correspondances
Pont Jacques-Cartier	170
Pont de Québec	158
Pont Victoria	156
Pont Champlain	154
Pont d'île d'Orléans	151
...	...

