

Chapitre 7 :

Extraction de primitives

Partie 1 – Détection de coins

Joël Lefebvre (UQÀM)

INF600F – Traitement d'images

Automne 2024

Sommaire

- Partie 1 du chapitre sur l'extraction de caractéristiques
- Motivation
- Détection de coins
- Détecteur de coins de Harris

Références

- Gonzalez, Ch12 : *Feature Extraction*
- Szeliski, Ch4 : *Feature detection and matching*
- Burger, Vol2, Ch4 : *Corner Detection*

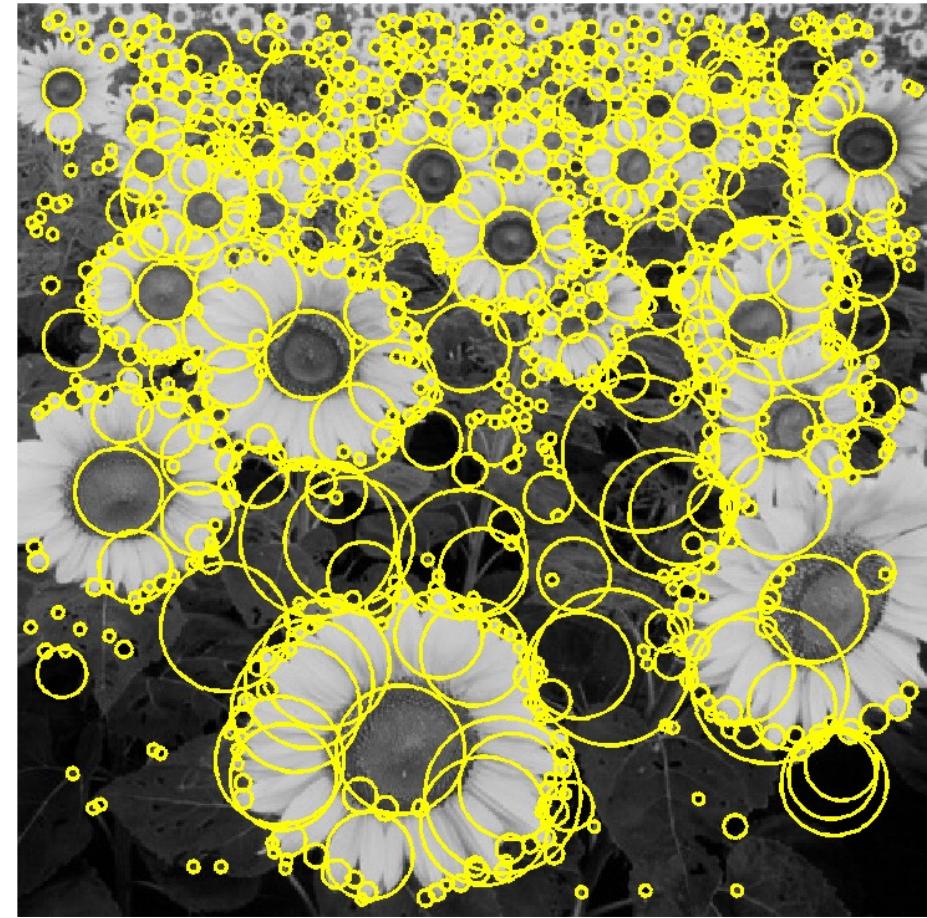
Détection de coins

Chapitre 7, Partie 1 – Détection de coins

Joël Lefebvre (UQÀM)

INF600F – Traitement d'images

Détection de coins et de taches



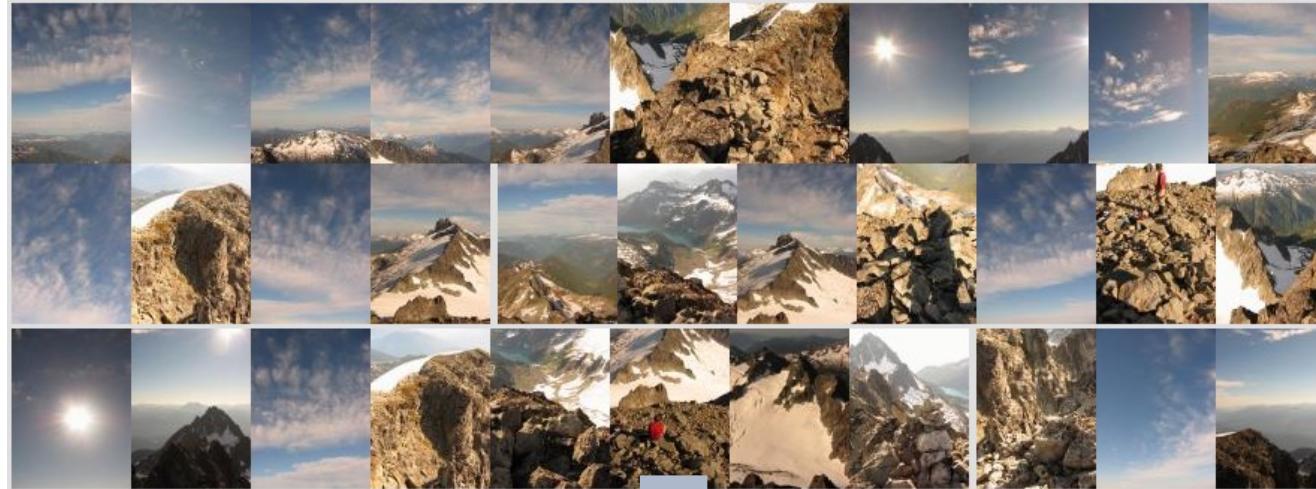
(Source : N.₆ Snavely)

Contexte global

- Points caractéristiques (*feature points*)
- Aussi nommés points d'*intérêt* (*interest points*), points clés (*key point*), etc. Parfois décrits comme des caractéristiques « locales »



Motivation : Panorama automatique

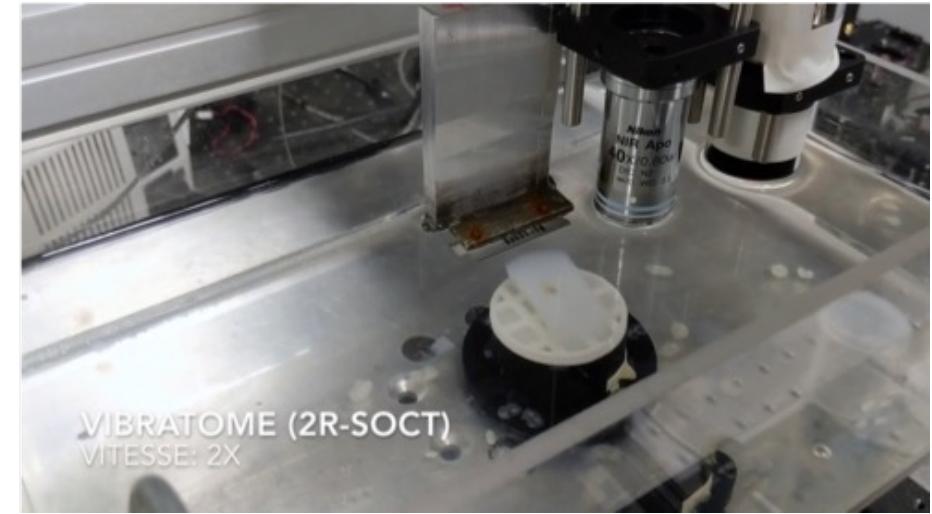
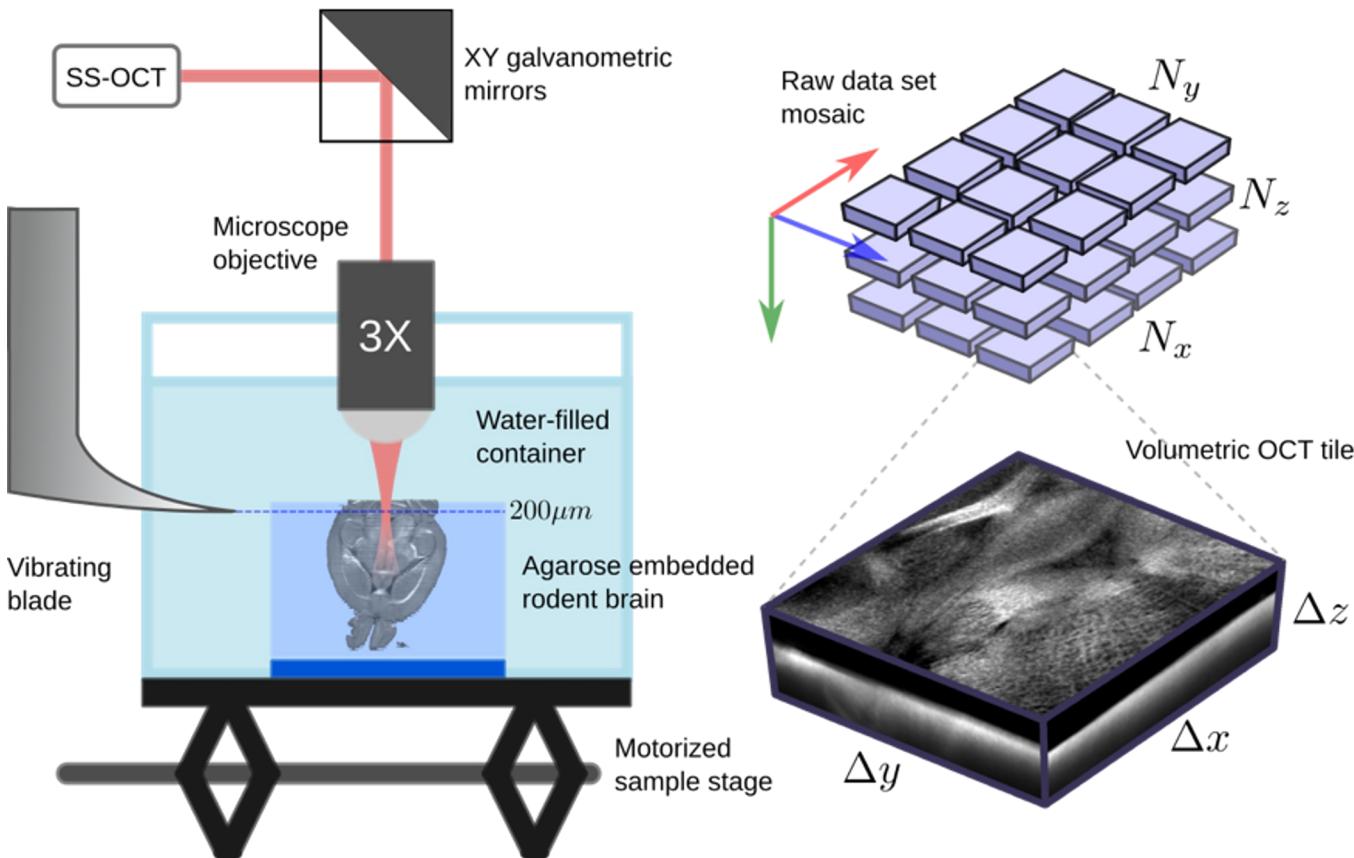


(Source : N. Snavely)

Credit: Matt Brown

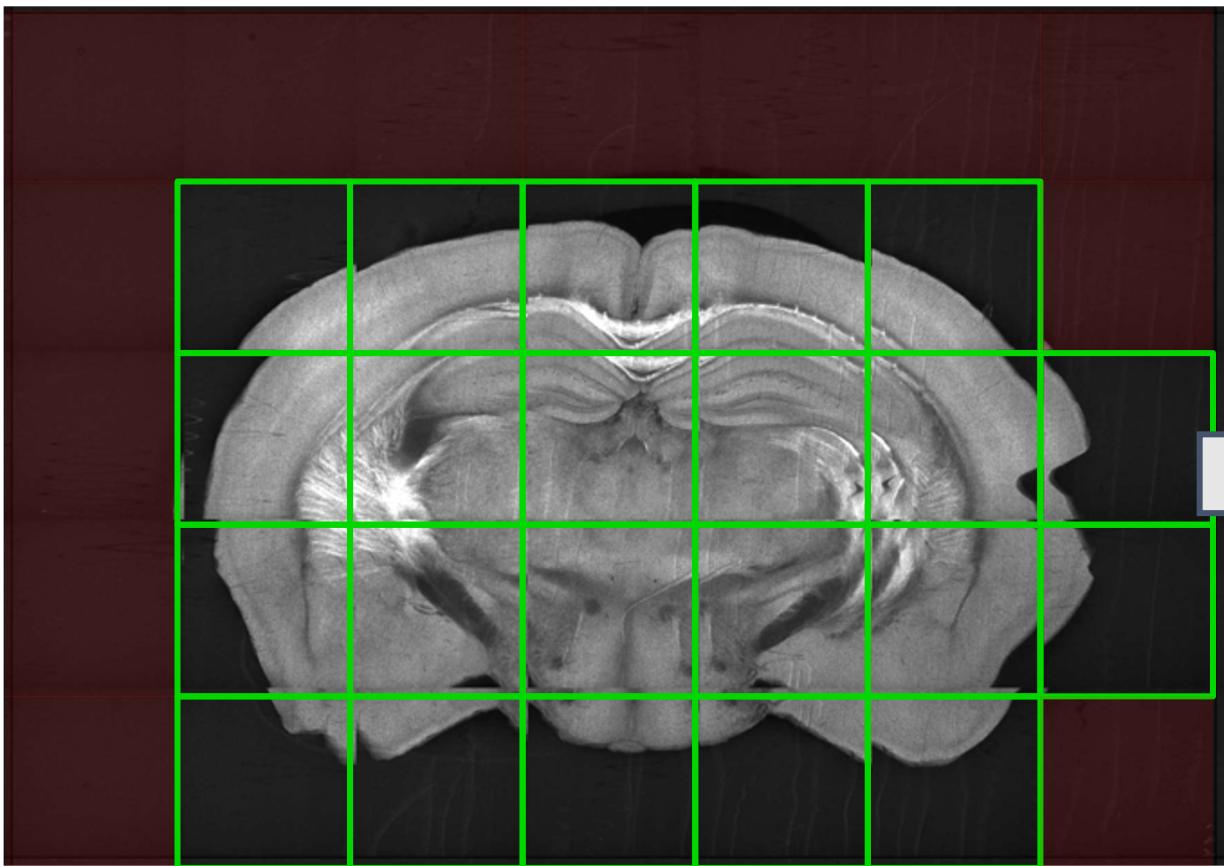


Motivation : Histologie sérielle automatique



Contactez-moi si vous êtes intéressés à faire un stage d'été, une maîtrise ou un doctorat en lien avec le traitement d'images, la vision par ordinateur, l'apprentissage profond et la microscopie / l'imagerie numérique

Motivation : Histologie sérielle - II

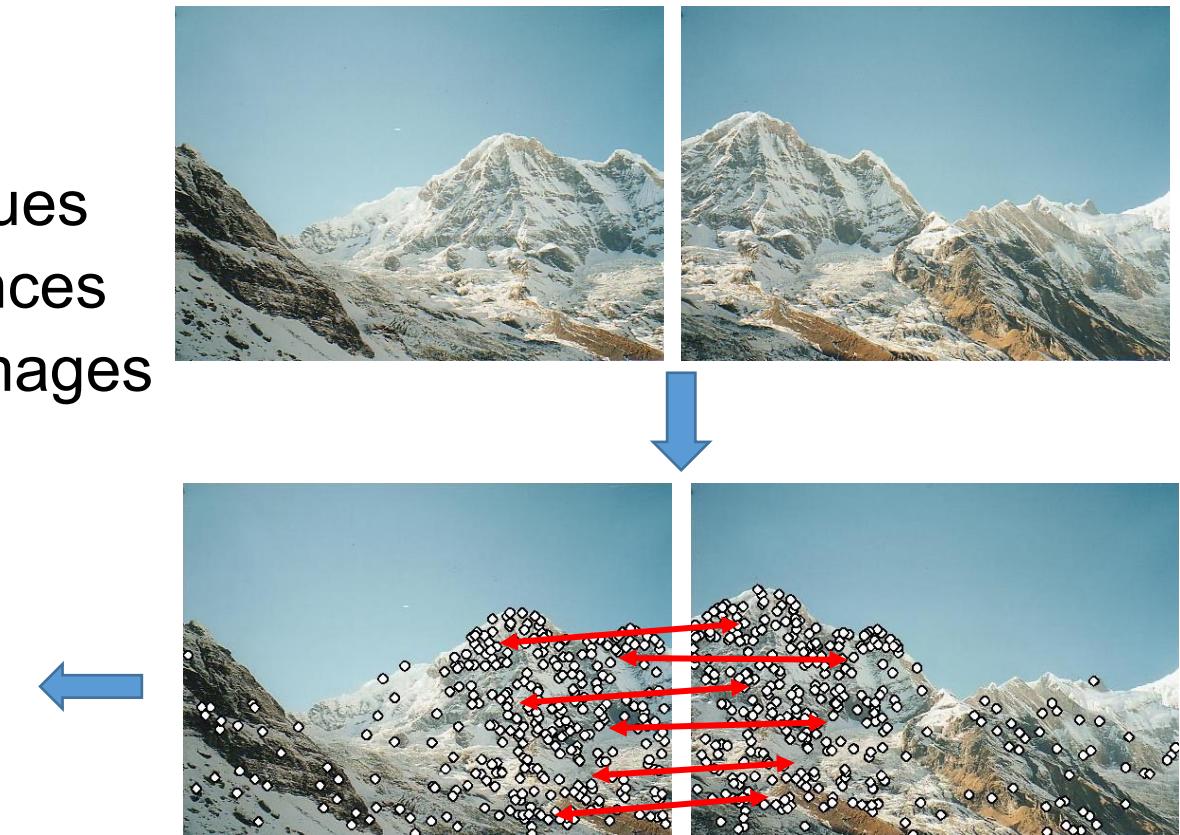


Recalage et assemblage d'une mosaïque d'images



Pourquoi extraire des caractéristiques ?

- **Exemple** : Reconstruction de panoramas
 - Étape 1 : Extraire des caractéristiques
 - Étape 2 : Trouver les correspondances
 - Étape 3 : Aligner et fusionner les images

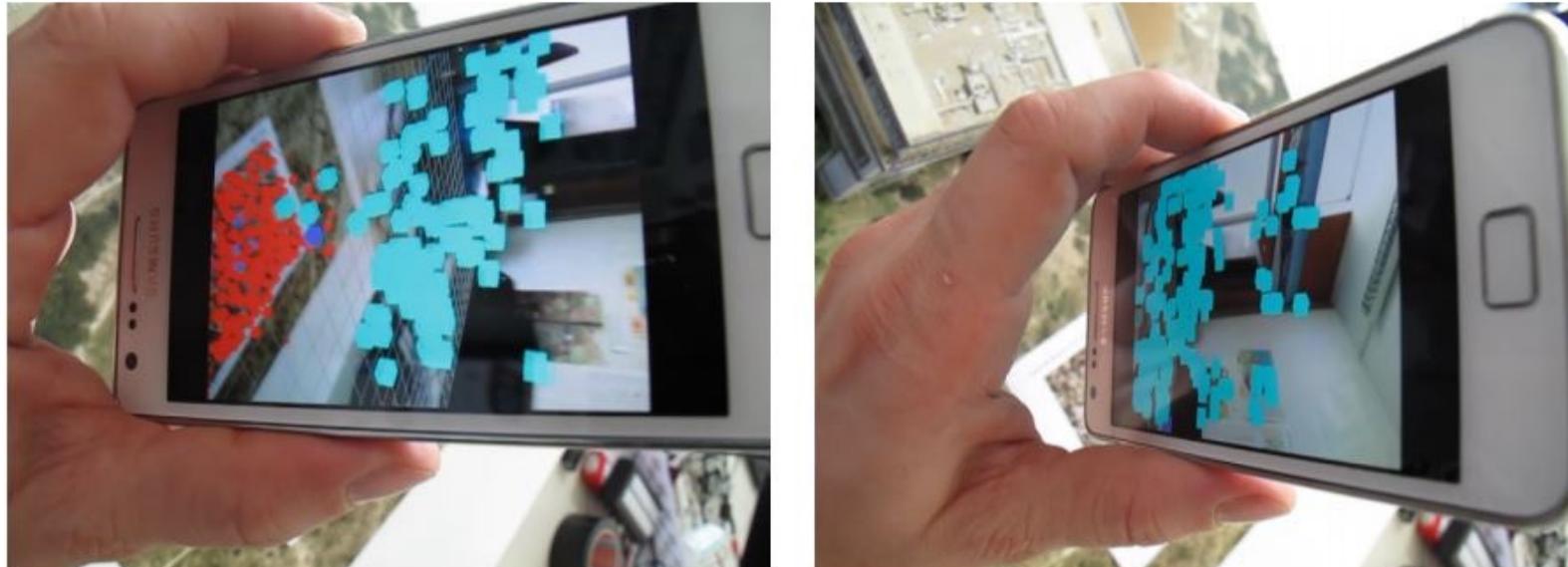


(Source : N. Snavely)

Application : SLAM Visuel

- SLAM : *Simultaneous localization and mapping*
- Création et mise à jour d'une carte d'un environnement inconnu, tout en naviguant dans cet environnement
- Applications en robotique, en réalité virtuelle, réalité augmentée, voiture autonome, drone de livraison ou d'urgence ...

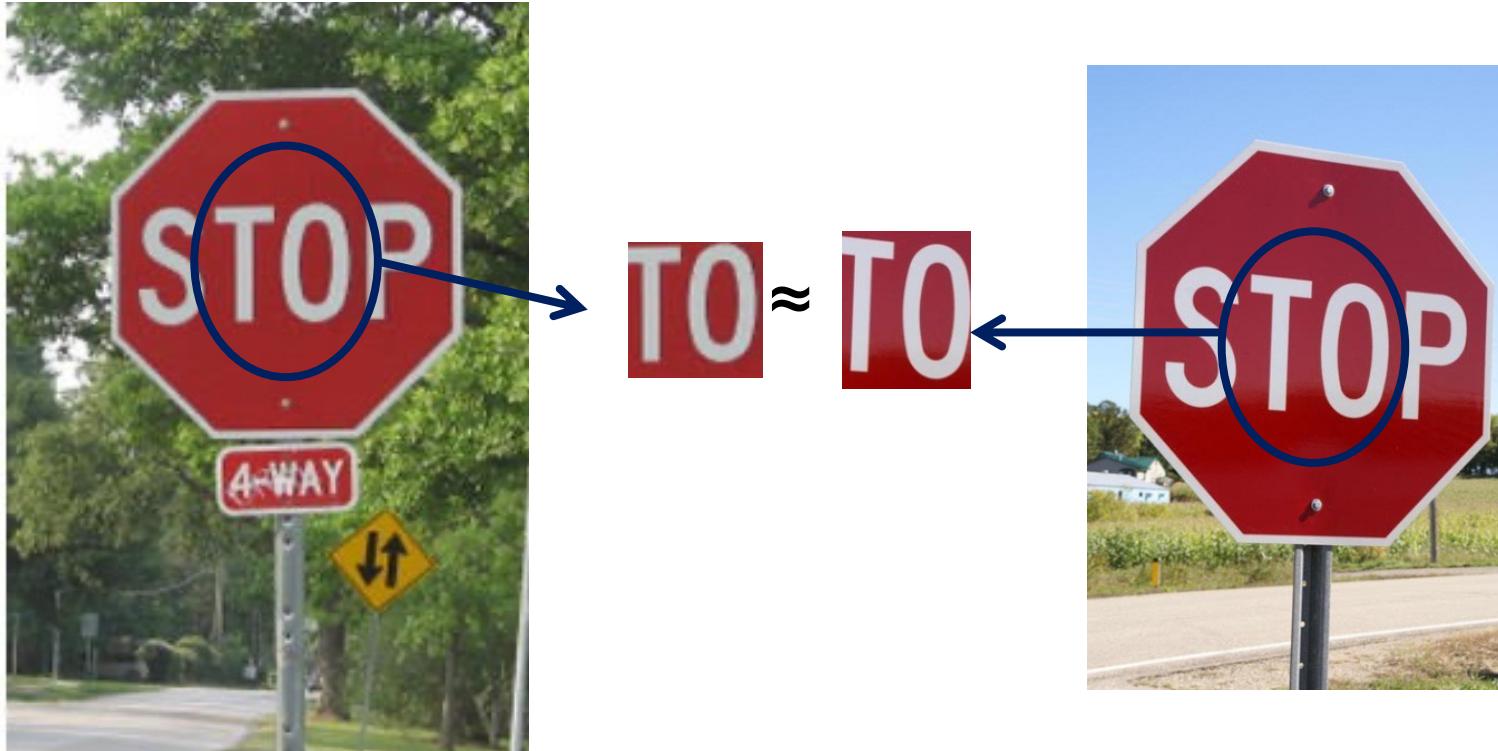
(Source : N. Snavely)



Application : Correspondance d'images

Détection d'objets similaires

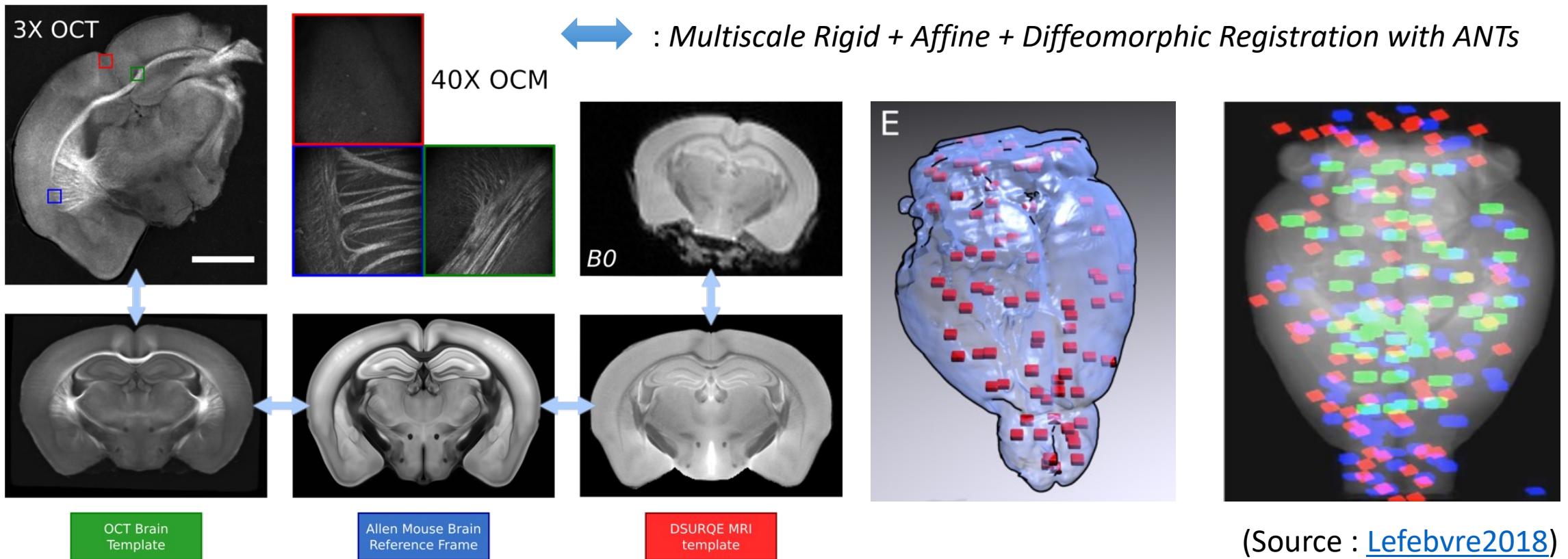
- Correspondance : Paireage de points, de sous-régions, de contours ou de régions entières entre plusieurs images



Source : Malik & Hays

Application : Correspondance d'images (1)

- Exemple : Identifier les structures du cerveau présentes dans l'image



(Source : [Lefebvre2018](#))

Application : Correspondance d'images (2)

- Exemple : Trouver une image similaire dans une base de données



by [Diva Sian](#)



by [swashford](#)

Fontaine de
Trevi à Rome
[Wikipedia](#)

(Source : N. Snavely)

Correspondance d'images : Complexité (1)

- Source de complexité : Rotation de la scène, changement d'illumination



by [Diva Sian](#)



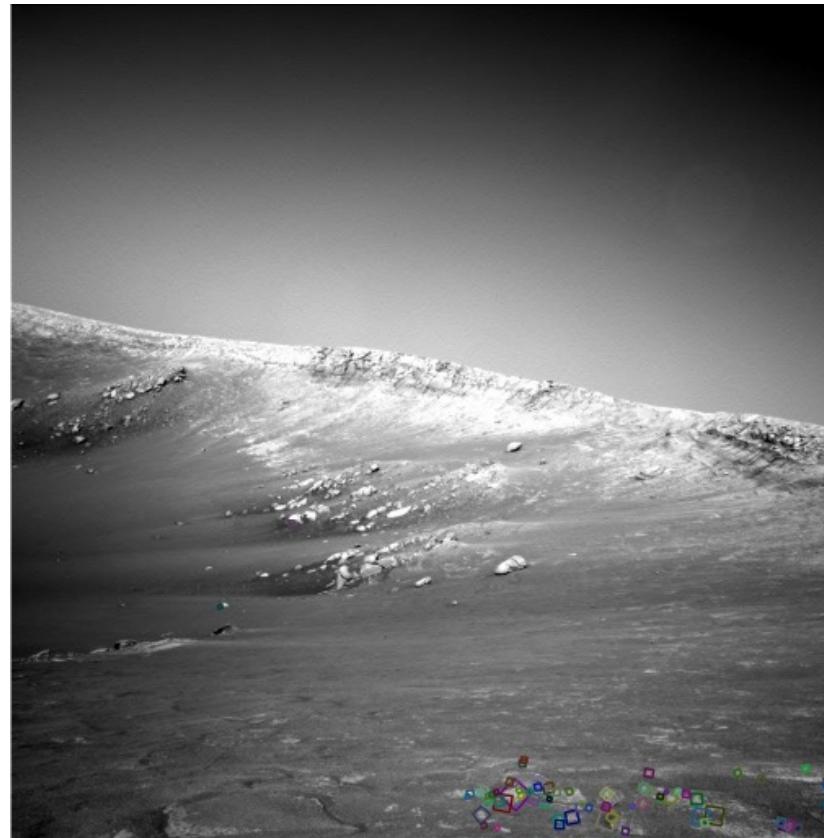
Fontaine de Trevi à Rome
[Wikipedia](#)

by [scgbt](#)

(Source : N. Snavely)

Correspondance d'images : Complexité (2)

- Source de complexité : Déplacement de la caméra



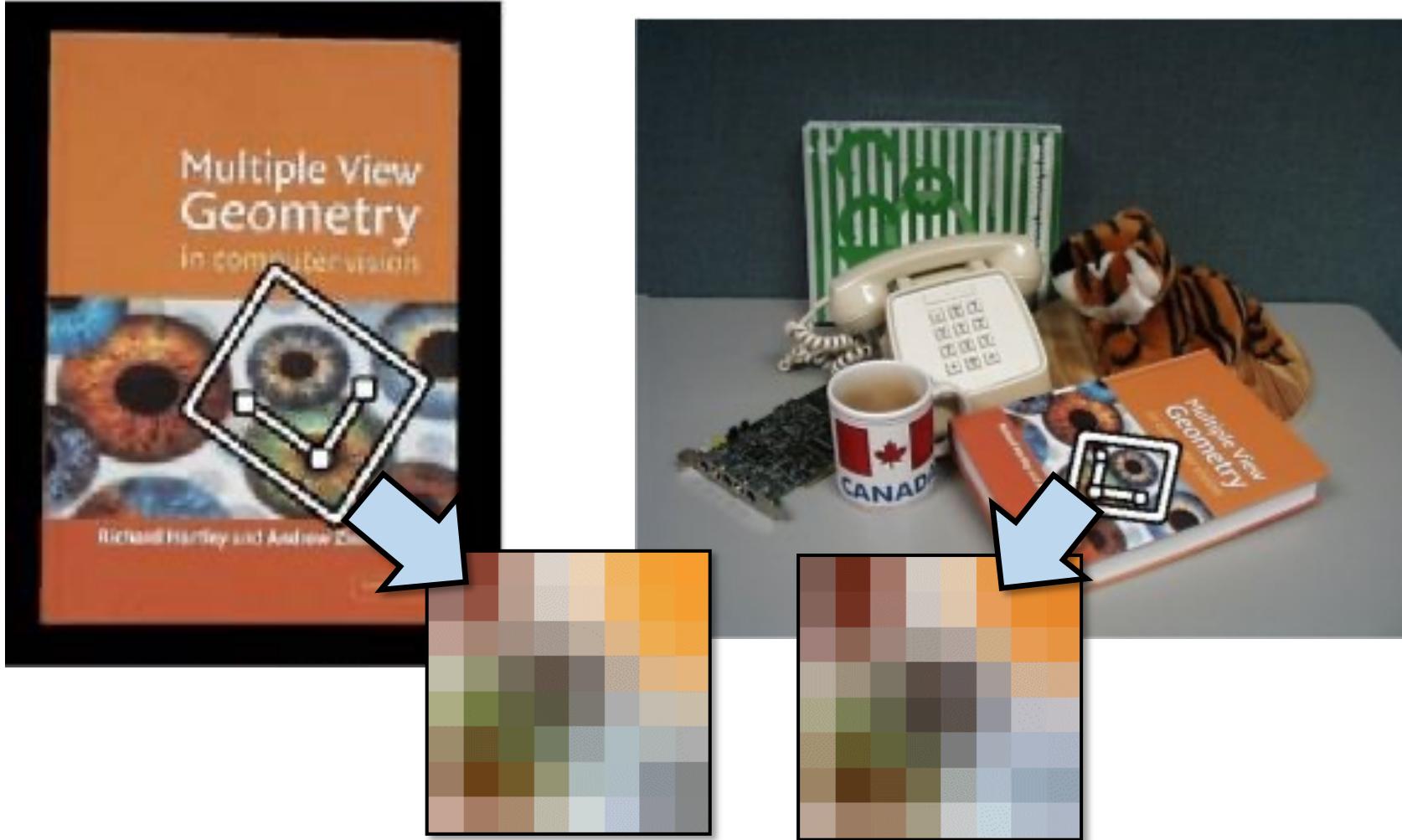
Images de la surface de la planète Mars provenant de la caméra du robot Curiosity

Où sont les points communs aux 2 images ?

Voir les petits carrés colorés détectés par la méthode SIFT

(Source : N. Snavely)

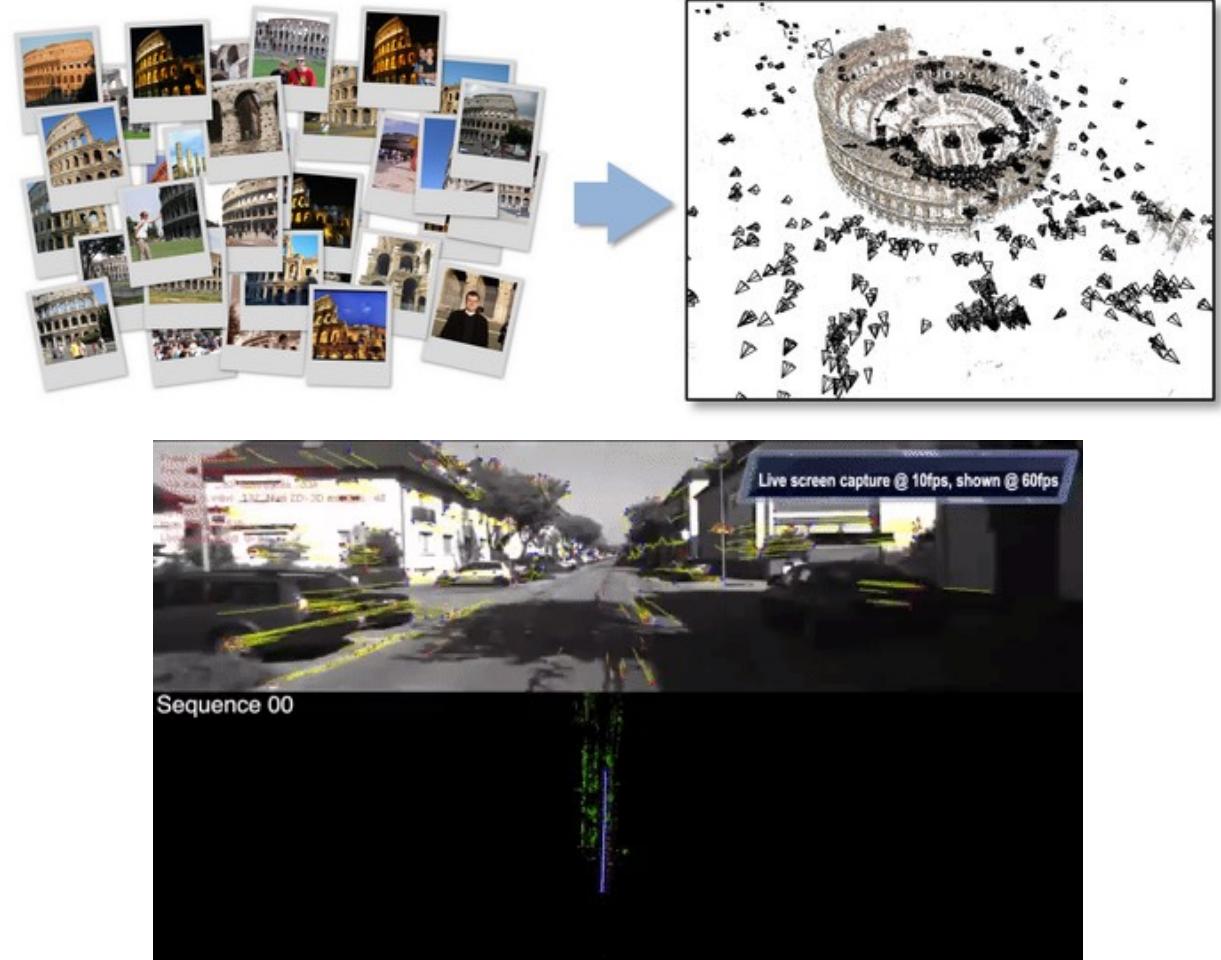
Application : Correspondance de caractéristiques (*Feature Matching*)



(Source : N. Snavely)
18

Quelques autres applications ...

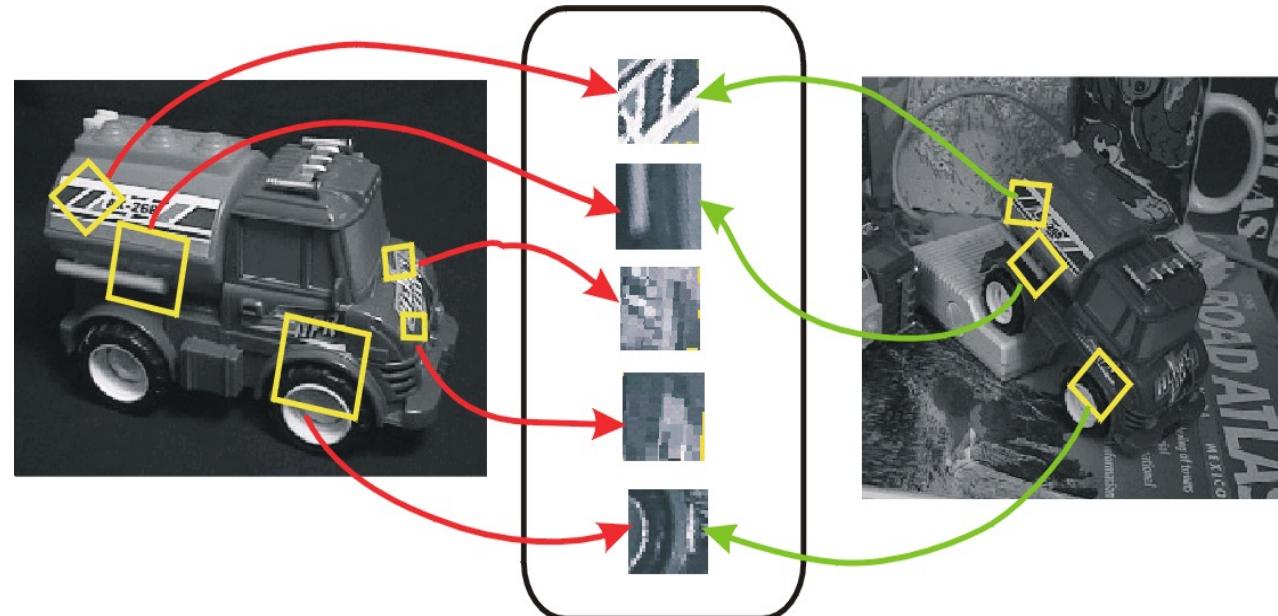
- Alignement d'images
- Reconstruction 3D
- Suivi de mouvements
- Reconnaissance d'objets
- Recherche d'images
- Navigation robotique
- Et plusieurs autres



(Source : N. Snavely)

Invariance des caractéristiques locales

- Déetecter des caractéristiques qui sont invariantes face à plusieurs transformations
- **Invariance géométrique** : Translation, Rotation, Échelle ...
- **Invariance photométrique** : Illumination, Exposition ...



(Source : N. Snavely)

Propriétés désirées des caractéristiques locales

- **Localisées** : Les caractéristiques sont locales, donc robustes face à l'occlusion et le désordre
- **Répétabilité** : Les mêmes caractéristiques peuvent être trouvées dans plusieurs images malgré leurs transformations géométriques et photométriques
- **Quantité** : Plusieurs centaines / milliers pour une seule image
- **Distinction / Saillance** : Permet de différencier une grande base de données d'objets
- **Efficacité** : Il y a beaucoup moins de points caractéristiques que de pixels / Possible d'obtenir des performances en temps réel

But : Répétabilité des opérateurs de détection des points d'intérêt

- On veut détecter les mêmes points dans chaque image

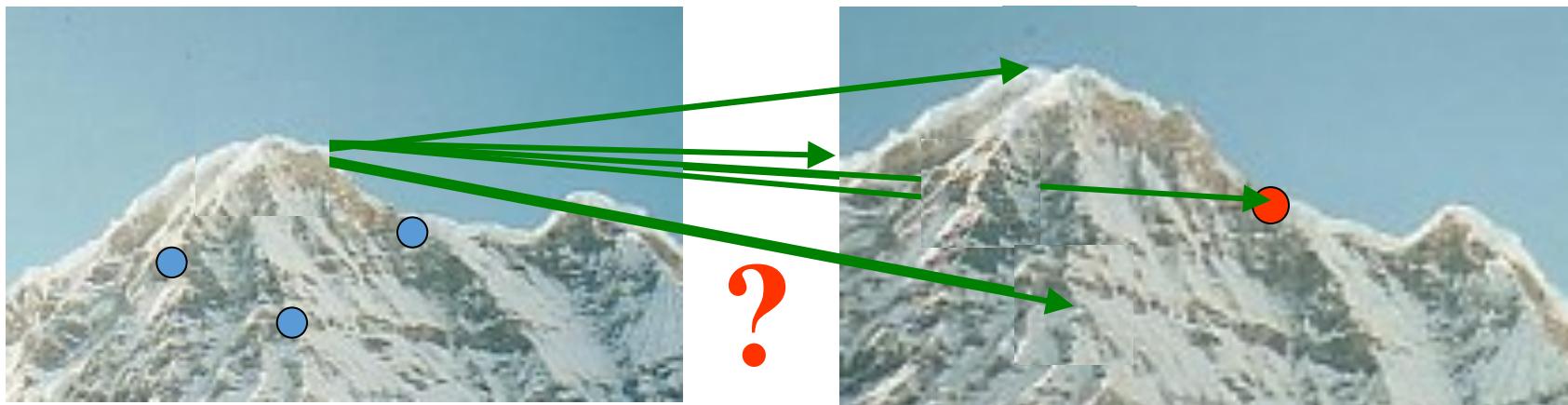


Avec ces points, il sera impossible de trouver une vraie correspondance entre ces images.

- On doit pouvoir être en mesure de détecter les points caractéristiques indépendamment pour chaque image.

But : Points d'intérêt distincts

- On veut pouvoir déterminer de façon fiable quels sont les points correspondants entre chaque image



- Doit être invariant face aux différences géométrique et photométrique entre plusieurs points de vue

Approche

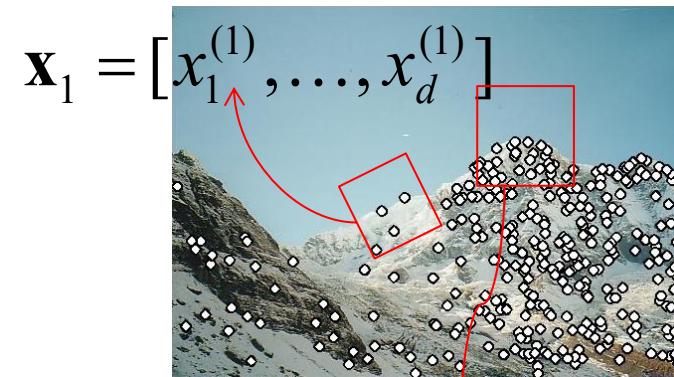
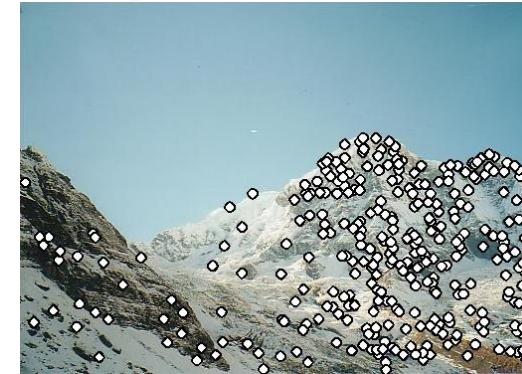
- Détection des caractéristiques (les trouver)
- Description des caractéristiques (les représenter)
- Correspondance des caractéristiques (les pairer)
- (*Optionnel*) Suivi de caractéristiques (les suivre lorsqu'elles sont en mouvement)

(Source : N. Snavely)

Caractéristiques locales : Composantes principales

- Détection : Identifier les points d'intérêt
- Description : Extraire un vecteur de caractéristiques (descripteur) autour de chaque point d'intérêt
- Pairage : Déterminer la correspondance entre les descripteurs des deux points de vue.

(Source : N .Snavely & Kristen Grauman)



$$\mathbf{x}_1 = [x_1^{(1)}, \dots, x_d^{(1)}]$$

$$\mathbf{x}_2 = [x_1^{(2)}, \dots, x_d^{(2)}]$$



Qu'est-ce qu'une bonne caractéristique?



[Source](#) : Dominick Gravel / Agence QMI

Détection : Idée de base

- On ne sait pas à priori avec quelle autre position dans l'image sera associée une caractéristique
- On peut toutefois calculer le niveau de stabilité d'une position quant à son apparence lorsque la position est légèrement modifiée
- *Comparaison des sous-régions (patch) avec son voisinage local*

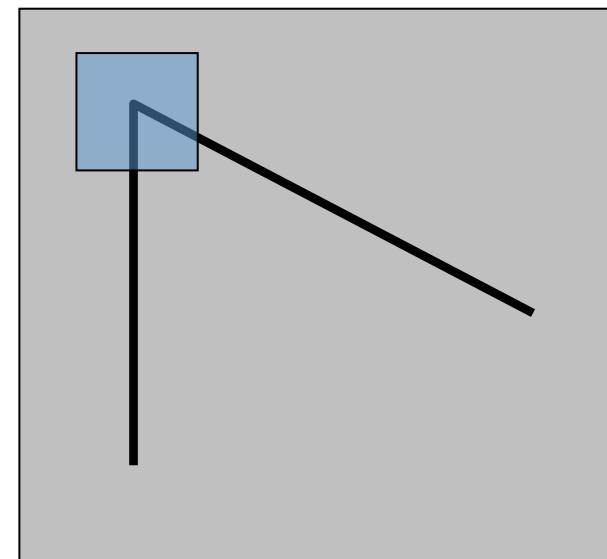
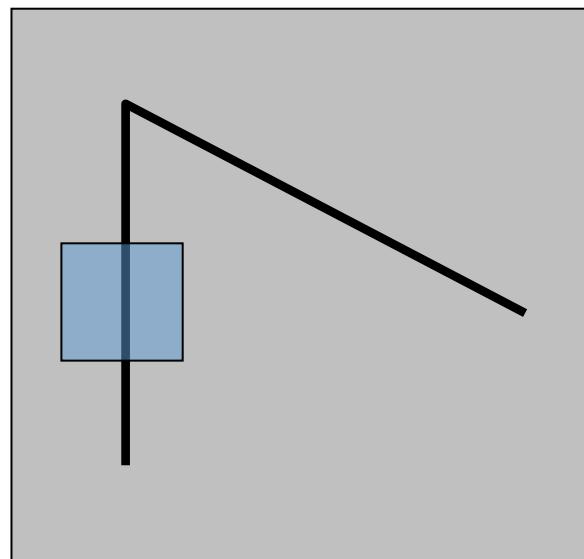
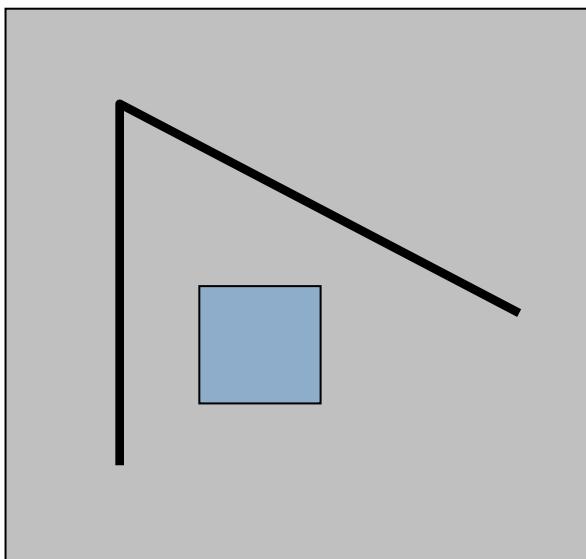


Unicité

- Les caractéristiques doivent être uniques
- Rechercher des régions de l'image qui sont non usuelles
- Résulte en des correspondances sans ambiguïté avec les autres images
- **Comment définir une région « non usuelle » ?**

Mesures locales de l'unicité (1)

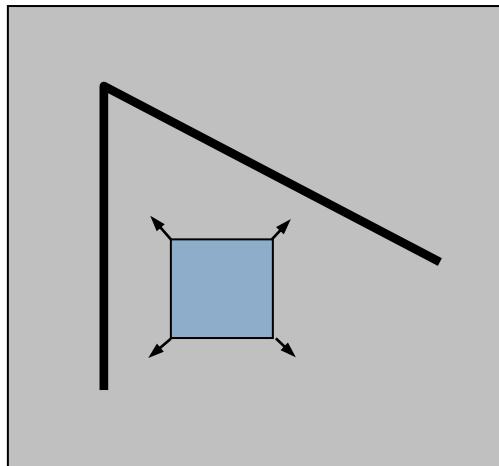
- Supposons qu'on ne considère qu'une petite fenêtre de pixels
- **Qu'est-ce qui fait qu'un candidat de caractéristique est bon ou mauvais?**



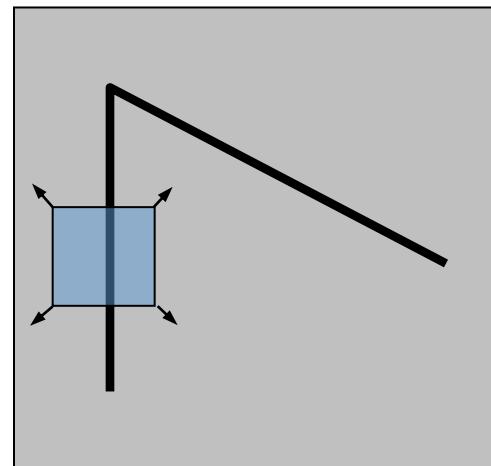
Mesures locales de l'unicité (2)

- Comment est-ce que les fenêtres changent lorsqu'elles se déplacent ?
- **Une bonne caractéristique** : Déplacer la fenêtre, peu importe la direction, cause un grand changement.

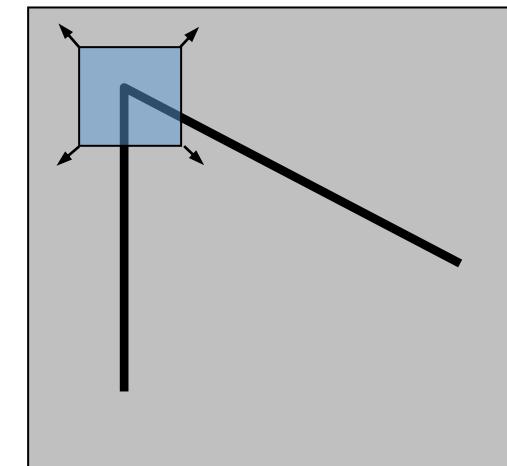
Credit: S. Seitz, D. Frolova, D. Simakov



Région « plate » : aucun changement dans toutes les directions



« Bordure » : Aucun changement dans la direction de la bordure



« Coin » : Changement significatif dans toutes les directions

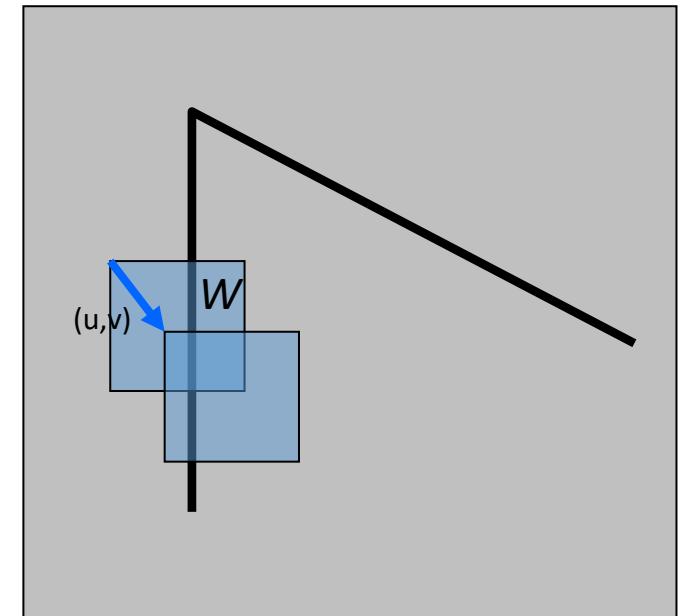
Détecteur de coins de Harris (*maths 1*)

Considérons la translation d'une fenêtre W par (u, v)

- Comment changent les pixels dans W ?
- Comparons chaque pixel avant et après le déplacement en additionnant leurs différences au carré (SSD pour *sum of squared differences*)
- Définissons « l'erreur » SSD

$$\begin{aligned} E(u, v) &= \sum_{(x,y) \in W} [I(x + u, y + v) - I(x, y)]^2 \\ &= \sum_{(x,y)} w(x, y) [I(x + u, y + v) - I(x, y)]^2 \end{aligned}$$

Fenêtrage, Intensité déplacée, Intensité

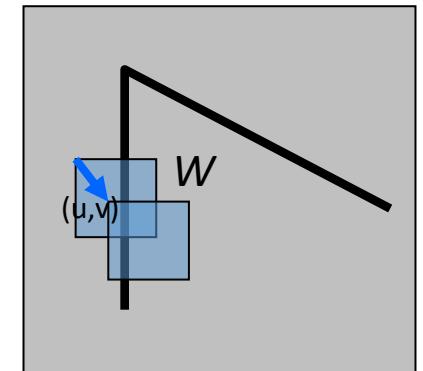


Détection de coins par auto-corrélation (1)

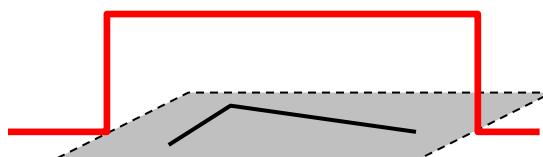
Changement de l'apparence dans la fenêtre $w(x, y)$ pour un déplacement $[u, v]$

$$E(u, v) = \sum_{(x,y)} w(x, y) [I(x + u, y + v) - I(x, y)]^2$$

Fenêtrage, Intensité déplacée, Intensité

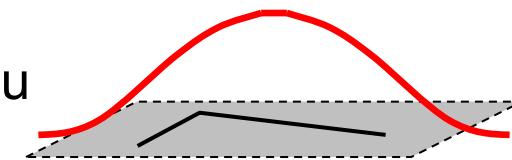


Fenêtrage $w(x, y) =$



1 dans la fenêtre, 0 ailleurs

ou



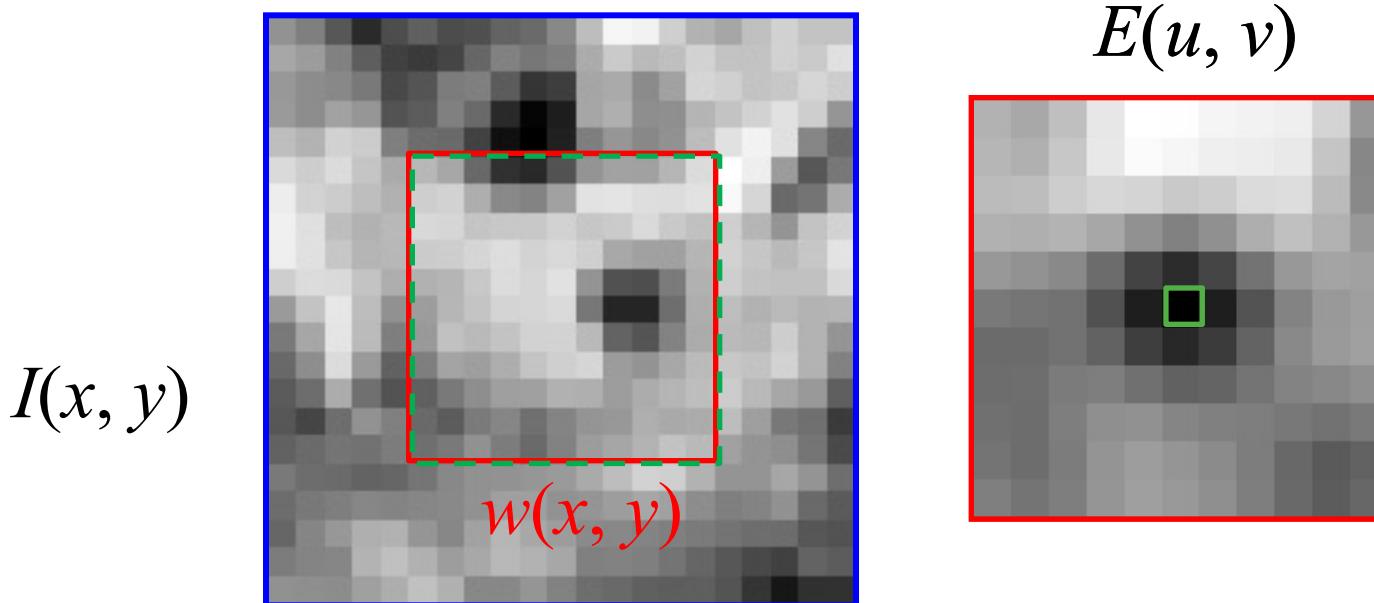
Gaussian

Source: R. Szeliski

Détection de coins par auto-corrélation (2)

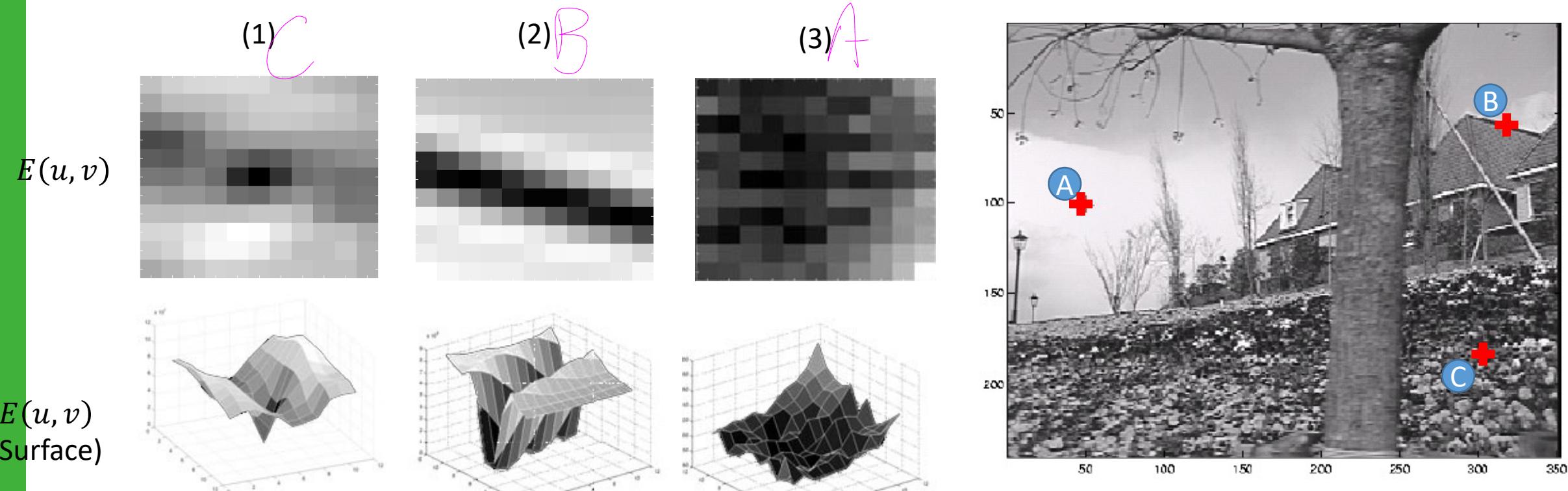
Changement de l'apparence dans la fenêtre $w(x, y)$ pour un déplacement $[u, v]$

$$E(u, v) = \sum_{(x,y)} w(x, y) [I(x + u, y + v) - I(x, y)]^2$$



Question : Associer les positions à leur fonction d'autocorrélation $E(u, v)$

- $E(u, v) = \sum_{(x,y)} w(x, y)[I(x + u, y + v) - I(x, y)]^2$



Détection de coins par autocorrélation (3)

- Changement de l'apparence dans la fenêtre $w(x, y)$ pour un déplacement $[u, v]$:
- $E(u, v) = \sum_{(x,y)} w(x, y)[I(x + u, y + v) - I(x, y)]^2$
- On désire calculer l'allure de $E(u, v)$ pour de petits déplacements
- Long à calculer directement
 - $\mathcal{O}(\text{taille fenêtre}^2 \times \text{étendue du déplacement}^2 \times \text{taille image}^2)$
 - $\mathcal{O}(11^2 \times 11^2 \times 600^2) = 5.2$ milliards de points
 - 14.6K par pixel de l'image



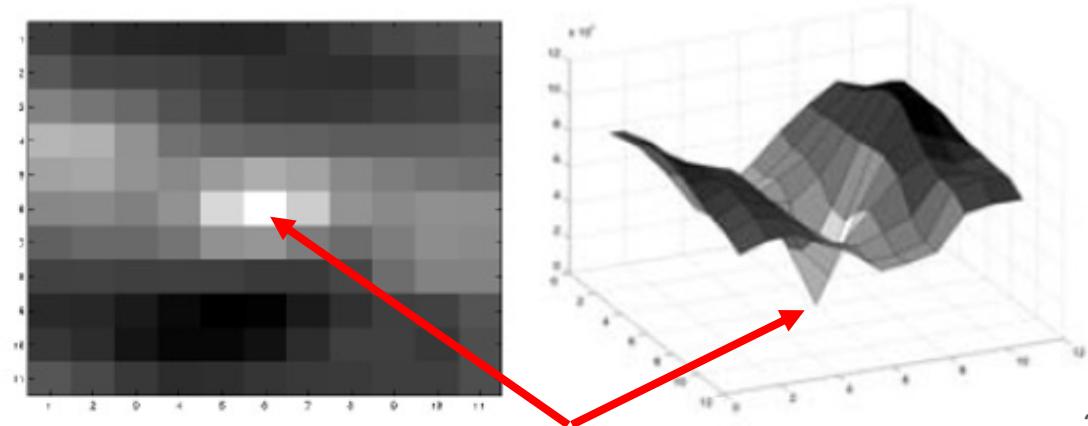
Détection de coins par autocorrélation (4)

Changement de l'apparence dans la fenêtre $w(x, y)$ pour un déplacement $[u, v]$:

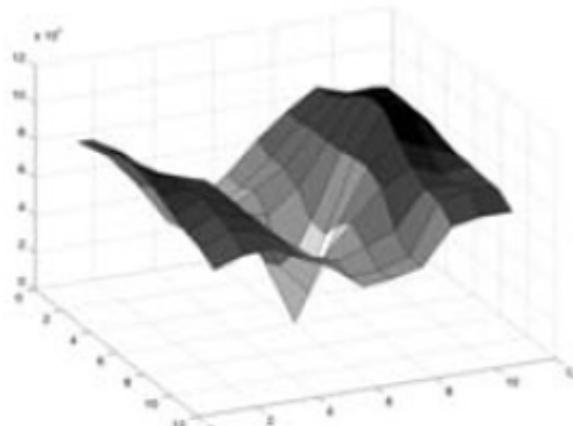
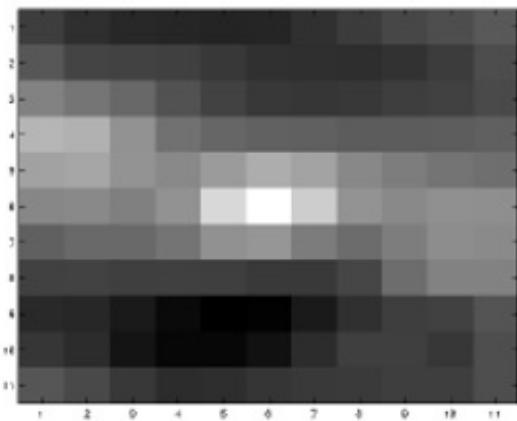
$$E(u, v) = \sum_{(x,y)} w(x, y) [I(x + u, y + v) - I(x, y)]^2$$

- On désire calculer l'allure de $E(u, v)$ pour de petits déplacements

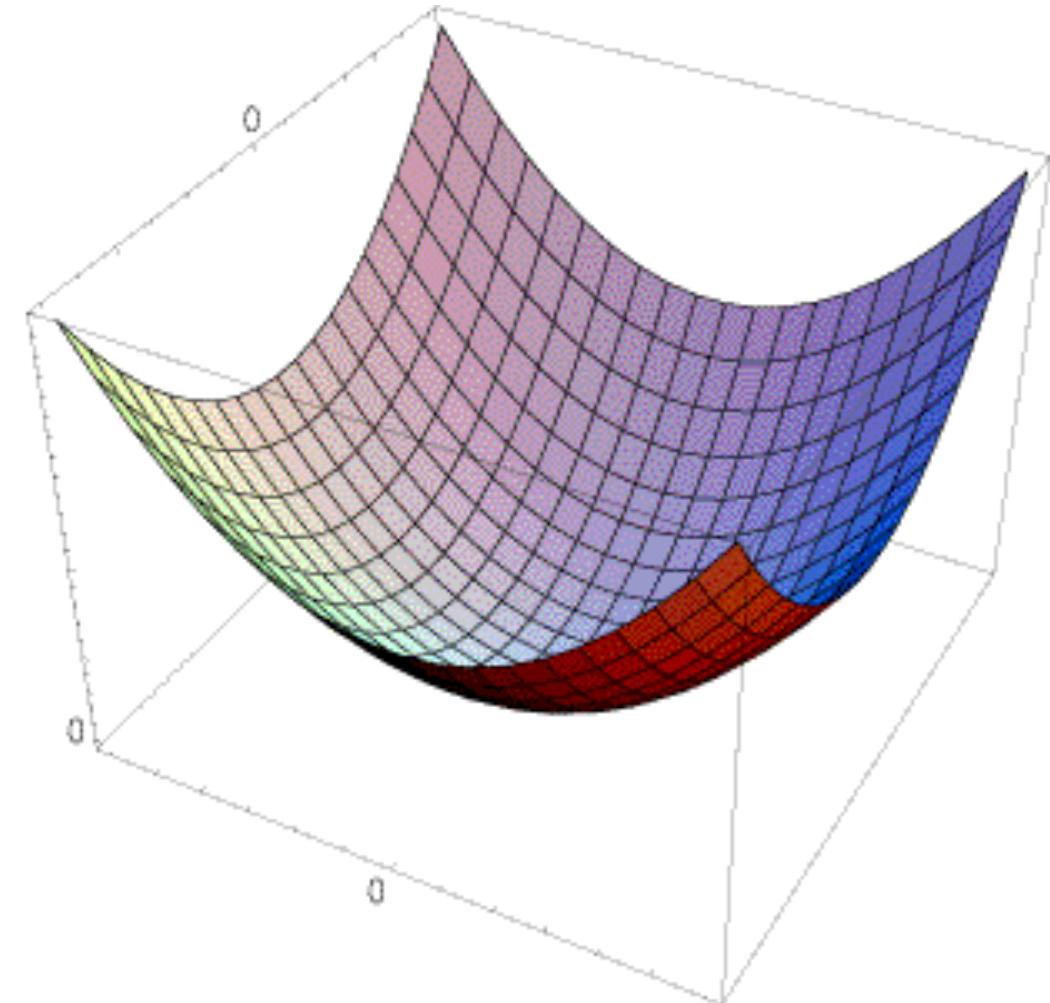
On connaît l'allure de $E(u, v)$ que l'on recherche : **sommet prononcé**



Approximation de $E(u, v)$ par une surface quadratique ?



\approx



Rappel : Expansion en série de Taylor

- Une fonction f peut être représentée par une série infinie de ses dérivées centrées sur le point a :

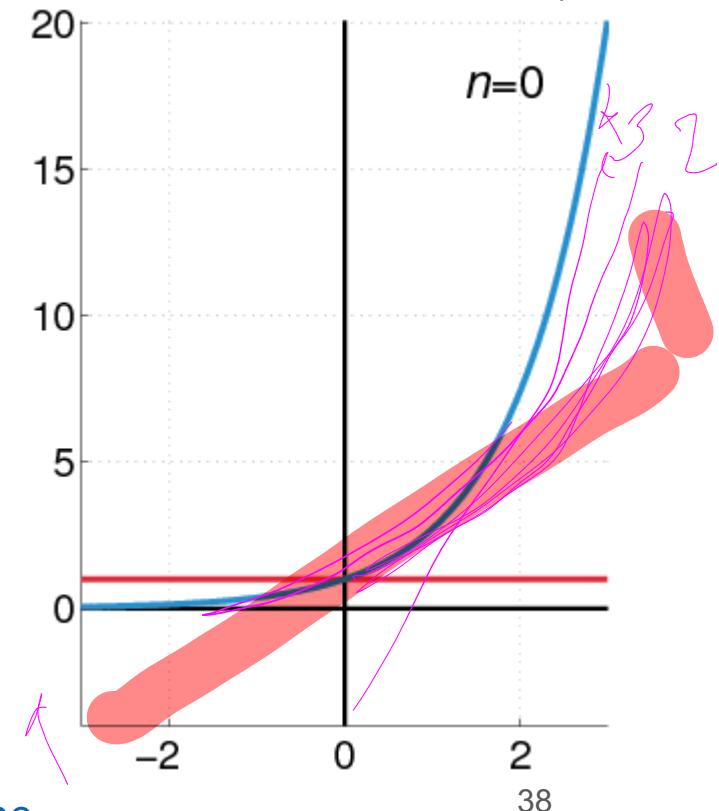
$$f(x) = \sum_{k=0}^{+\infty} \frac{f^{(k)}(a)}{k!} (x - a)^k$$

$$\begin{aligned} &= f(a) + \frac{f'(a)}{1!} (x - a) + \frac{f''(a)}{2!} (x - a)^2 + \frac{f'''(a)}{3!} (x - a)^3 \\ &\quad + \dots \end{aligned}$$

- Puisqu'on s'intéresse uniquement à la fenêtre centrée, on fixe $a = 0$ (Série de MacLaurin)

Approximation de
 $f(x) = e^x$ centrée en
 $f(0)$

Wikipedia



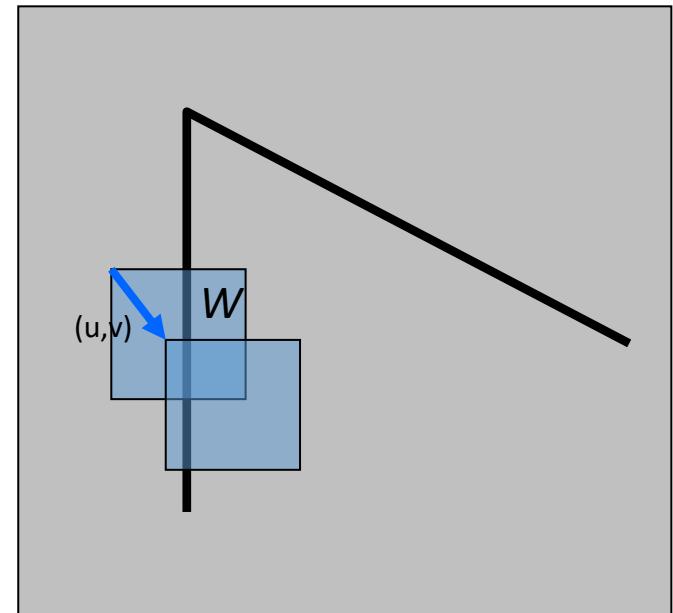
Hypothèse des petits déplacements

- Expansion en série de Taylor de l'image I :
- $I(x + u, y + v) = I(x, y) + \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v + \text{termes d'ordre supérieur}$
- Si le déplacement (u, v) est petit, l'approximation du 1^{er} ordre est suffisante
- $$\begin{aligned} I(x + u, y + v) &\approx I(x, y) + \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v \\ &\approx I(x, y) + [I_x \quad I_y] \begin{bmatrix} u \\ v \end{bmatrix} \end{aligned}$$
 Pour simplifier :
$$I_x = \frac{\partial I}{\partial x}$$
- Si on remplace ceci dans l'équation de l'erreur SSD ...

Détecteur de coins de Harris (maths 2)

- Considérons la translation d'une fenêtre W par (u, v)
- Définissons « l'erreur » SSD

$$\begin{aligned} E(u, v) &= \sum_{(x,y) \in W} [I(x + u, y + v) - I(x, y)]^2 \\ &\approx \sum_{(x,y) \in W} [I(x, y) + I_x u + I_y v - I(x, y)]^2 \\ &\approx \sum_{(x,y) \in W} [I_x u + I_y v]^2 \end{aligned}$$

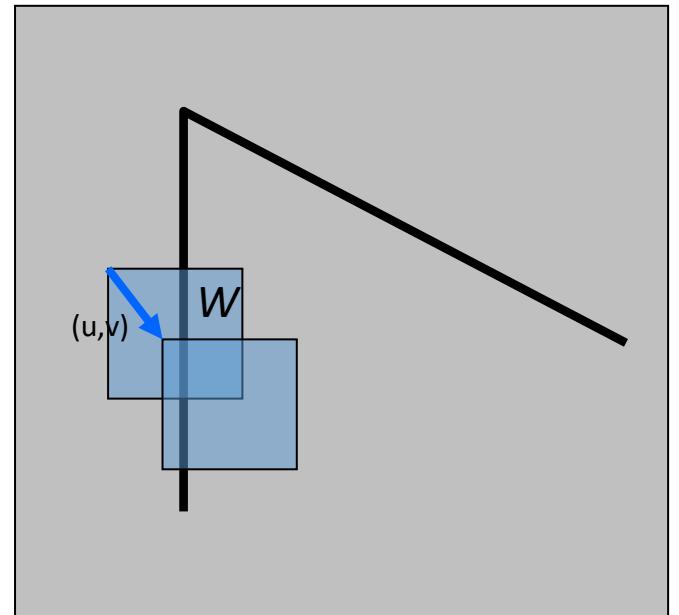


Détecteur de coins de Harris (maths 3)

- Considérons la translation d'une fenêtre W par (u, v)
- Définissons « l'erreur » SSD

$$\begin{aligned} E(u, v) &\approx \sum_{(x,y) \in W} [I_x u + I_y v]^2 \\ &\approx Au^2 + 2Buv + Cv^2 \end{aligned}$$

$$A = \sum_{(x,y) \in W} I_x^2 \quad B = \sum_{(x,y) \in W} I_x I_y \quad C = \sum_{(x,y) \in W} I_y^2$$



Donc, $E(u, v)$ peut localement être approximé par une fonction d'erreur quadratique

Tenseur de structure (Matrice du second moment)

- La surface $E(u, v)$ peut localement être décrite par une forme quadratique
- $E(u, v) \approx Au^2 + 2Buv + Cv^2$

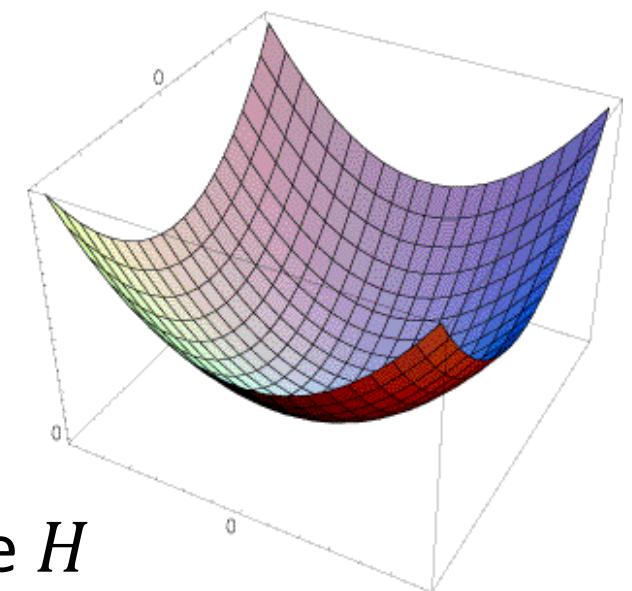
$$A = \sum_{(x,y) \in W} I_x^2$$

$$\approx [u \quad v] \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$B = \sum_{(x,y) \in W} I_x I_y$$

$$C = \sum_{(x,y) \in W} I_y^2$$

Tentons de comprendre H



Exemple de tenseur de structure

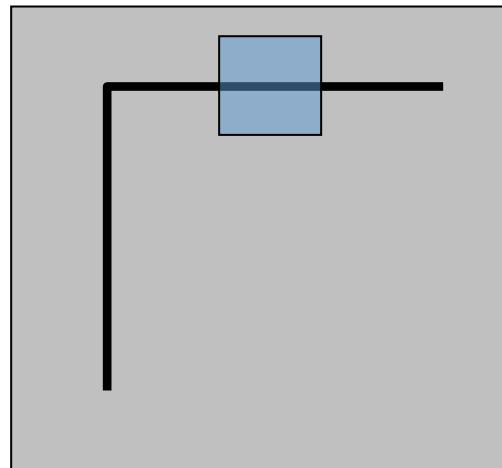
Contour horizontal

$$E(u, v) \approx [u \quad v] \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

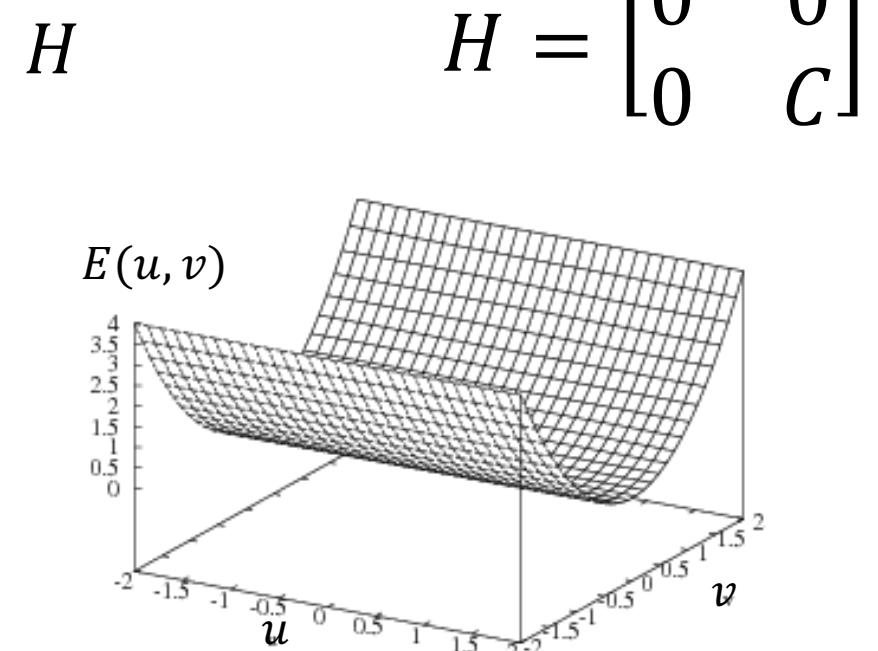
$$A = \sum_{(x,y) \in W} I_x^2$$

$$B = \sum_{(x,y) \in W} I_x I_y$$

$$C = \sum_{(x,y) \in W} I_y^2$$



Contour horizontal: $I_x = 0$



Exemple de tenseur de structure

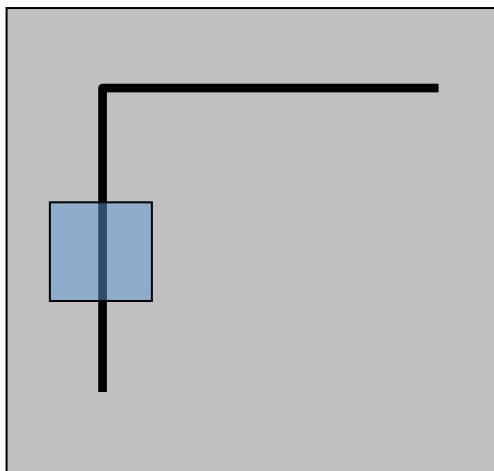
Contour vertical

$$E(u, v) \approx [u \quad v] \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y) \in W} I_x^2$$

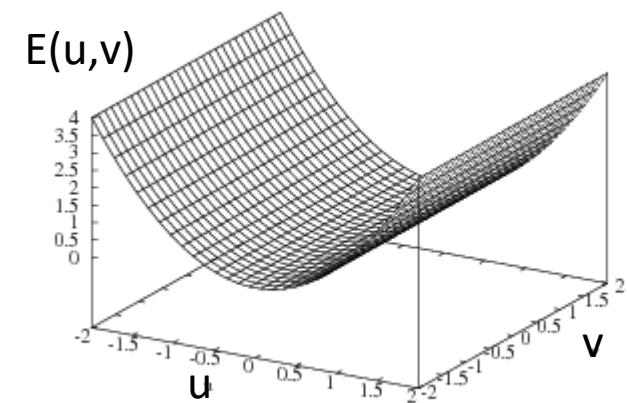
$$B = \sum_{(x,y) \in W} I_x I_y$$

$$C = \sum_{(x,y) \in W} I_y^2$$



Contour vertical : $I_y = 0$

$$H = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$$

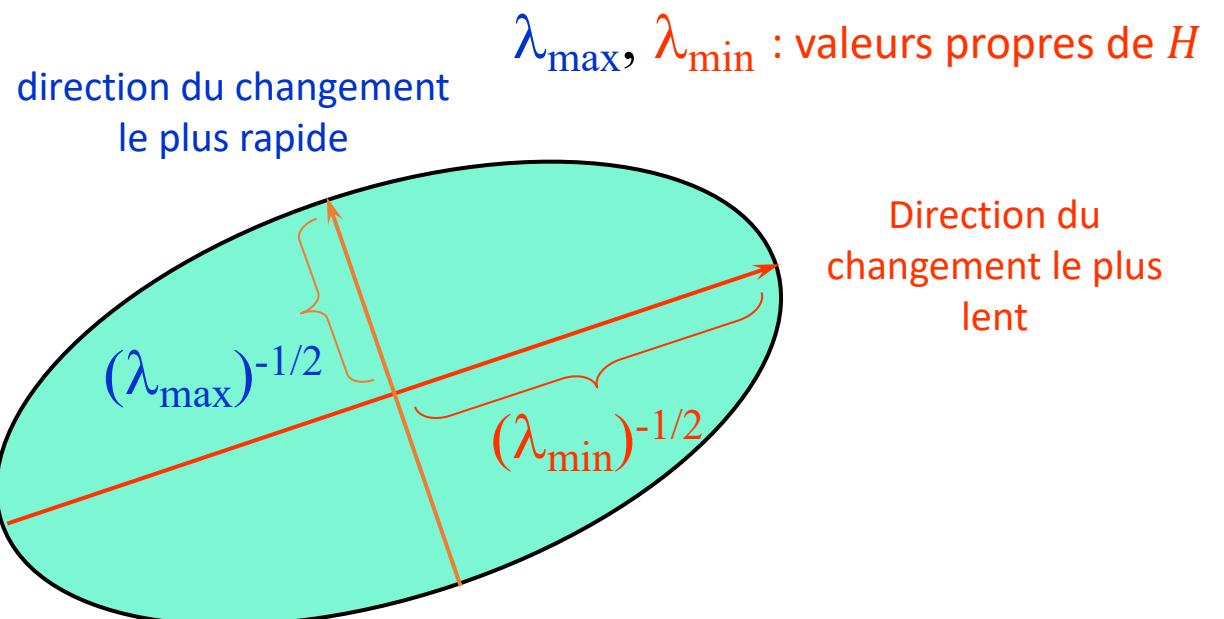
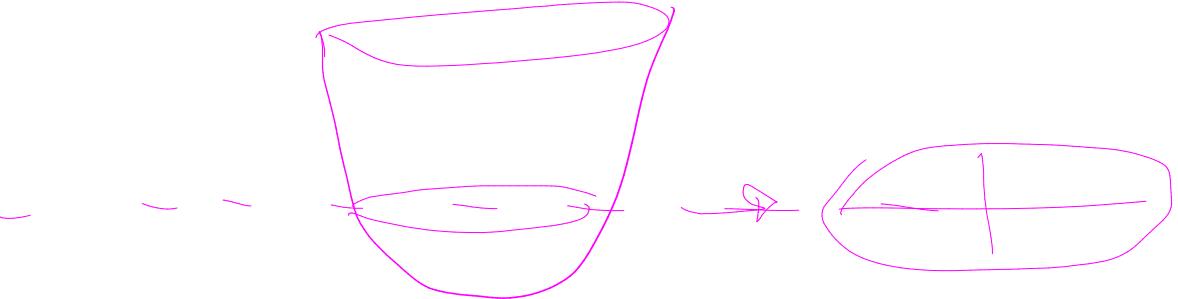


Cas général

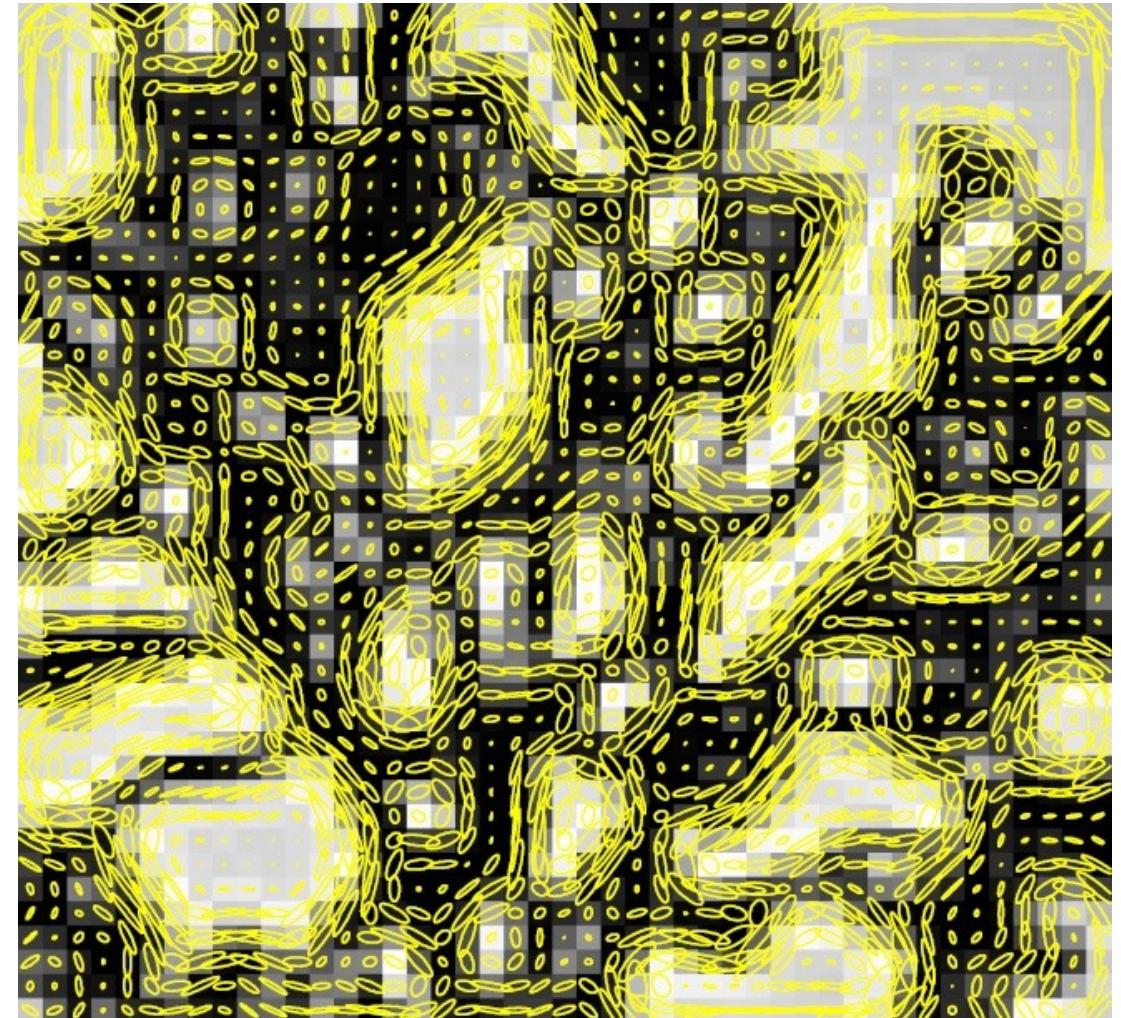
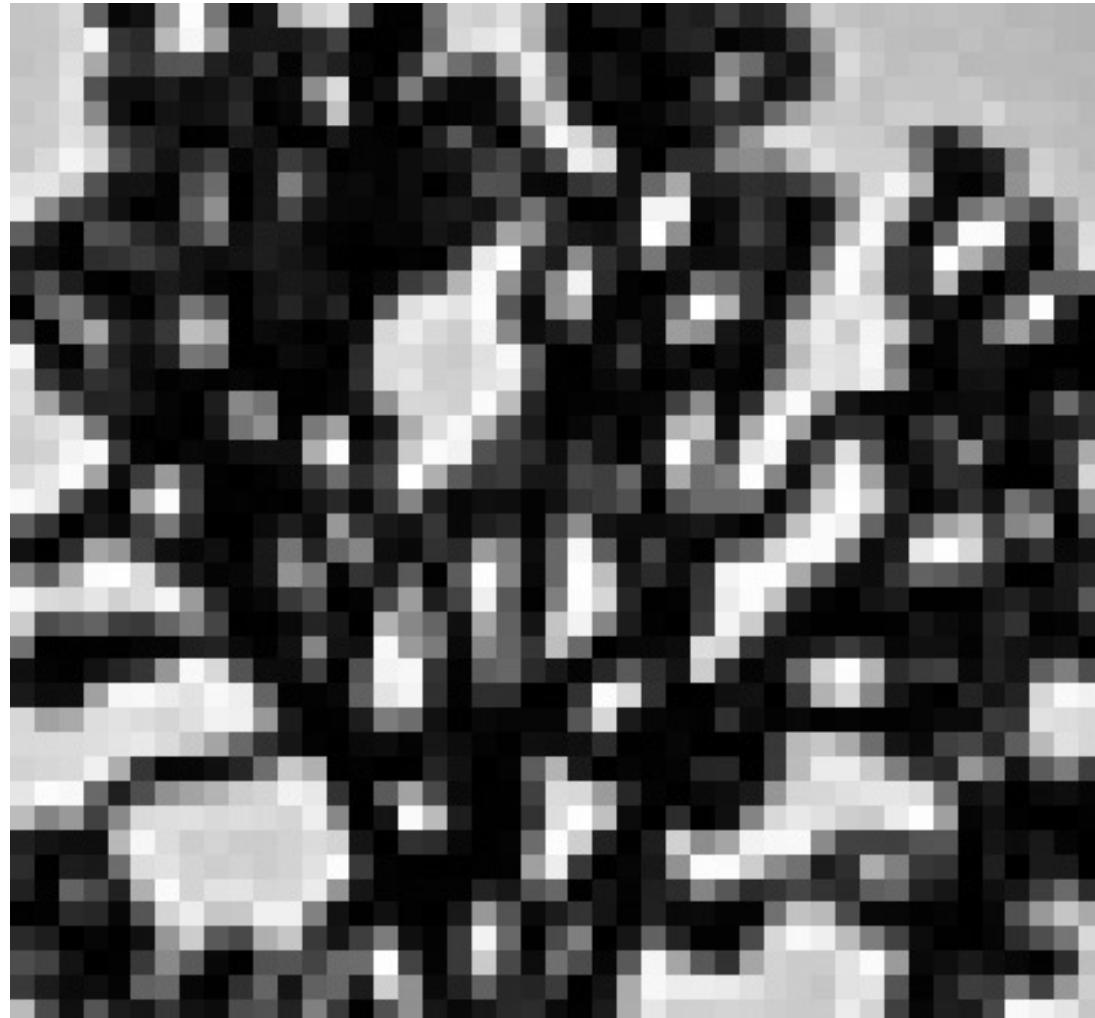
- On peut visualiser H comme étant une ellipse dont la longueur des axes est déterminée par les *valeurs propres* de H et dont l'orientation des axes est déterminée par les *vecteurs propres* de H

Équation d'un ellipse :

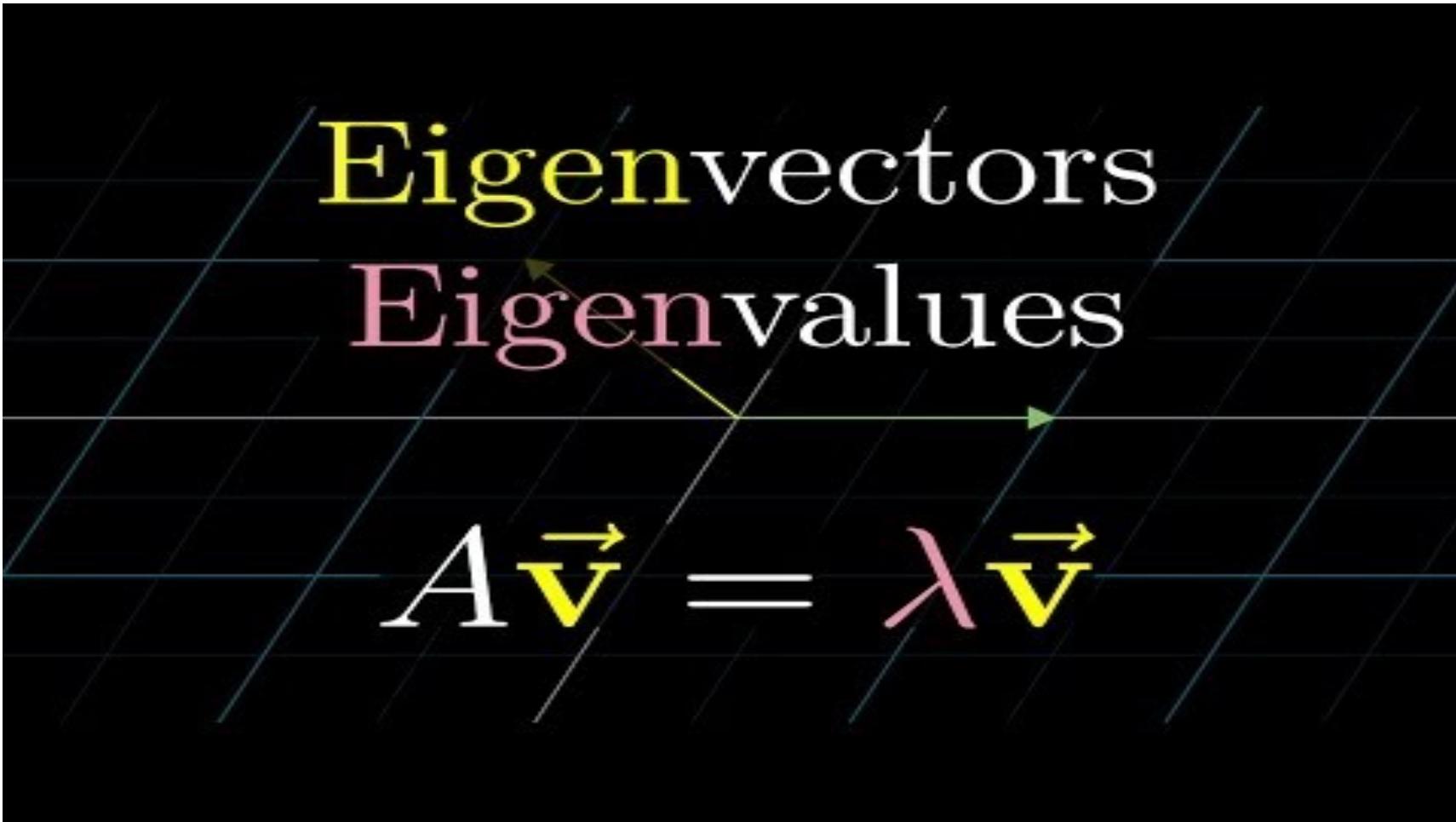
$$[u \ v]H \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$



Visualisation des tenseurs de structure



Rappel : Valeurs / Vecteurs propres (1)



Rappel : Valeurs / Vecteurs propres (2)

- Les **vecteurs propres** d'une matrice A sont les vecteurs x satisfaisant l'équation : $Ax = \lambda x$
- Le scalaire λ est la **valeur propre** correspondant à x
- Les valeurs propres sont trouvées en résolvant : $\det(A - \lambda I) = 0$
- Pour le tenseur de structure, $A = H$ est une matrice 2×2

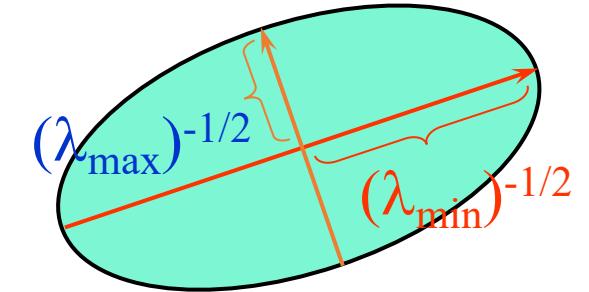
$$\det \begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} = 0$$

- La solution est : $\lambda_{\pm} = \frac{1}{2} \left[(h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2} \right]$
- Lorsque λ est connu, on pour trouver x en résolvant :

$$\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0$$

Détection de coins

Valeurs / Vecteurs propres 1



- Valeurs et vecteurs propres de H

- Définissent les directions de déplacement associées aux plus petit et plus grand changements de l'erreur $E(u, v)$
- x_{\max} : Direction de la plus grande augmentation de $E(u, v)$
- λ_{\max} : Grandeur de l'augmentation dans la direction x_{\max}
- x_{\min} : Direction de la plus petite augmentation de $E(u, v)$
- λ_{\min} : Grandeur de l'augmentation dans la direction x_{\min}

$$E(u, v) \approx [u \quad v] \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$
$$H$$

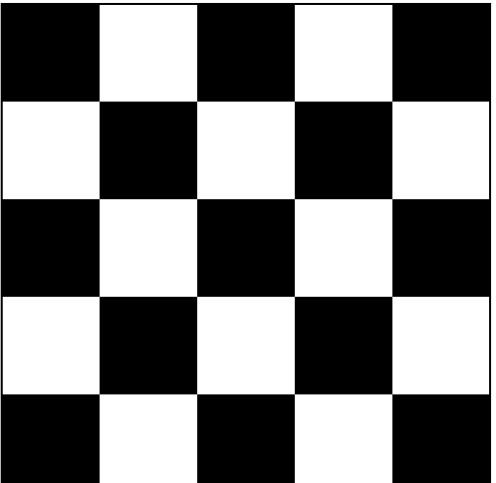
$$Hx_{\max} = \lambda_{\max}x_{\max}$$

$$Hx_{\min} = \lambda_{\min}x_{\min}$$

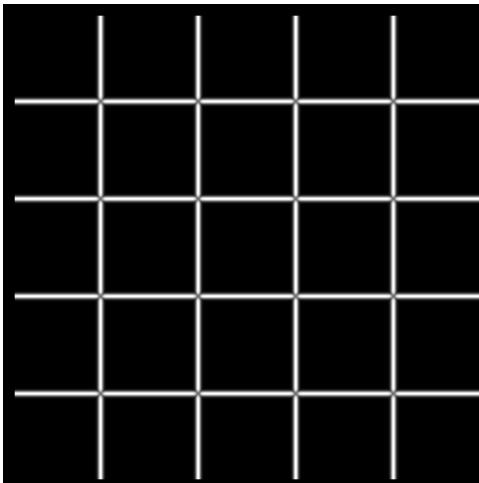
Détection de coins

Valeurs / Vecteurs propres 2

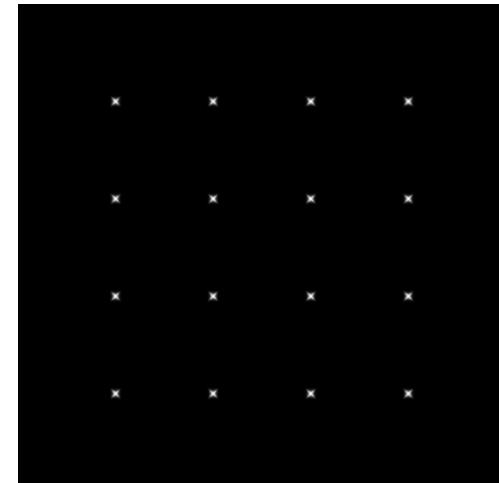
- Pourquoi λ_{max} , x_{max} , λ_{min} et x_{min} sont-ils importants pour la détection de caractéristiques ?
- Quelle est la fonction d'évaluation des caractéristiques ?
- On veut que l'erreur $E(u, v)$ soit grande pour de petits déplacements dans toutes les directions.
 - Le minimum de $E(u, v)$ devrait être grand, pour tous les vecteurs unitaires $[u, v]$
 - Le minimum est donné par la plus petite valeur propre λ_{min} de H



I



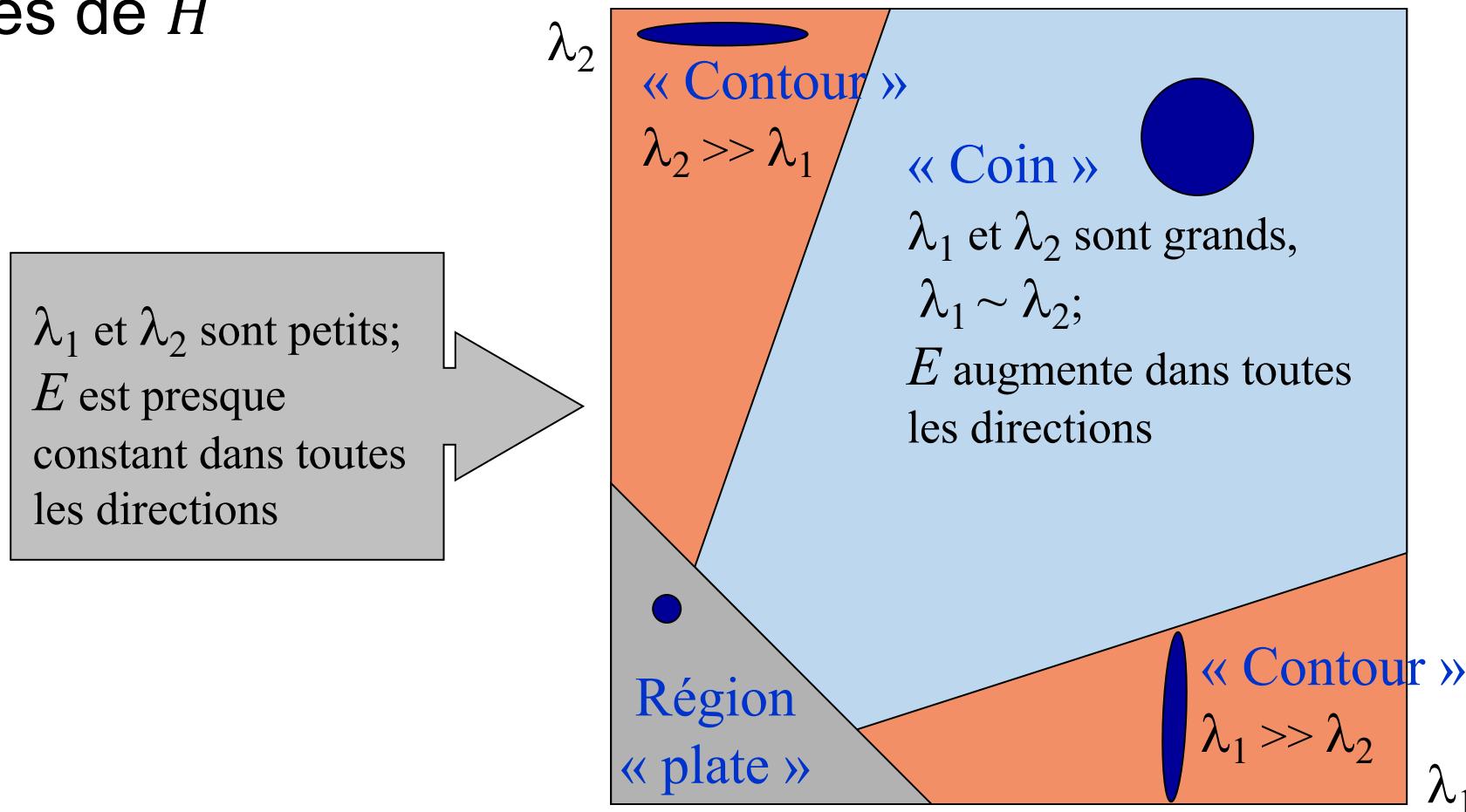
λ_{max}



λ_{min}

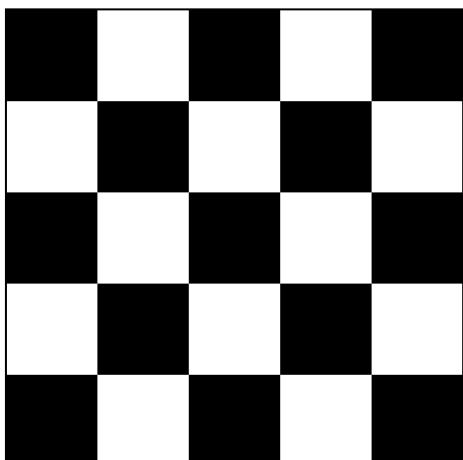
Interprétation des valeurs propres

- Classification des points de l'image en utilisant les valeurs propres de H

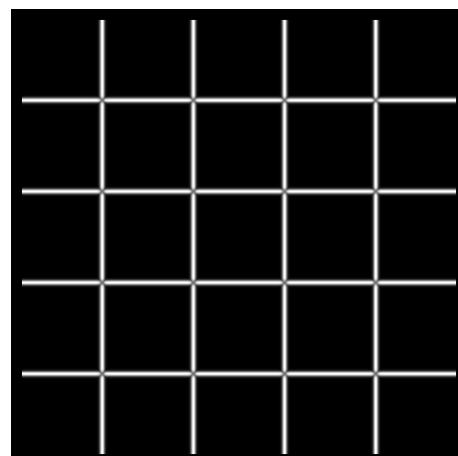


Résumé : Détection des coins

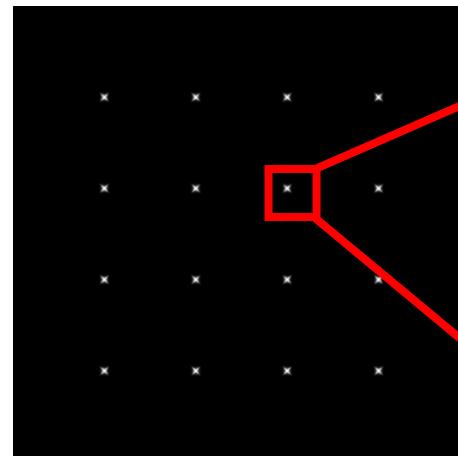
- Calcul du gradient en tout point de l'image
- Créer la matrice H à partir du gradient
- Calcul des valeurs propres de H
- Trouver les points avec une réponse élevée ($\lambda_{min} >$ seuil)
- Choisir en tant que caractéristiques les points pour lesquels λ_{min} est un maximum local



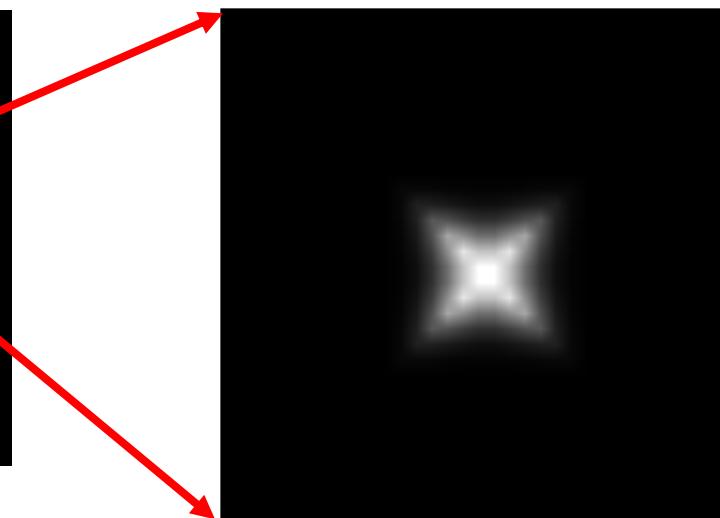
I



λ_{max}



λ_{min}



Classification des points de l'image basée sur les valeurs propres de H

- Exemple de métrique de similitude avec un coin (*cornerness*)

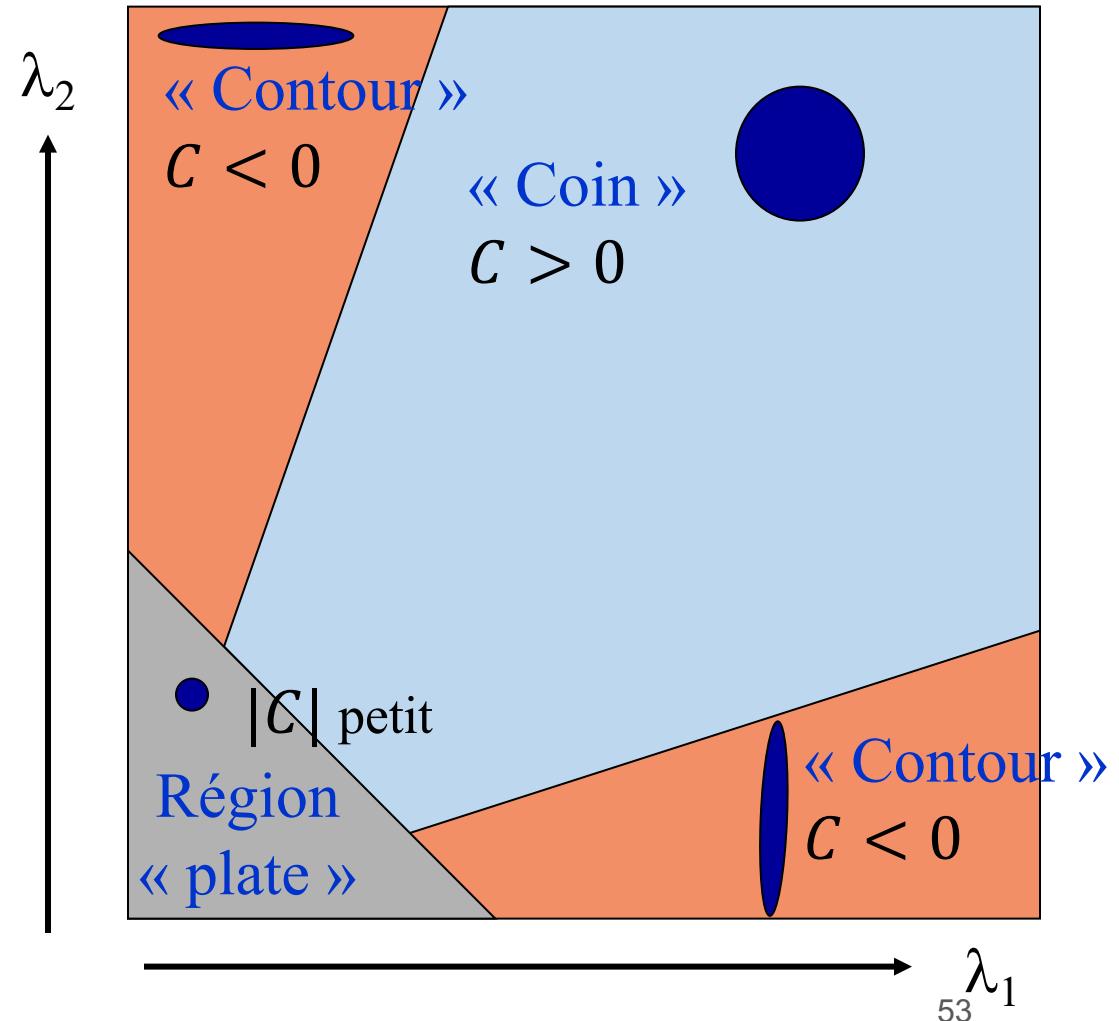
$$C = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

Avec α une constante

Rappel d'algèbre linéaire :

- Déterminant : $\det(A) = \prod_{i=1}^n \lambda_i = \lambda_1 \lambda_2 \cdots \lambda_n$
- Trace : $tr(A) = \sum_i \lambda_i$

$$C = \det(H) - \alpha \operatorname{trace}(H)^2$$



Détecteur de coins de Harris

- Calcul de la matrice H pour chaque fenêtre pour estimer la métrique de similitude au coin C (*cornerness score*)
- Seuillage pour trouver les pixels qui ont une forte réponse au détecteur de coins ($C > \text{seuil}$)
- Trouver les maximums locaux (c.-à-d. suppression des non-maximums)

C.Harris and M.Stephens. [“A Combined Corner and Edge Detector.”](#)
Proceedings of the 4th Alvey Vision Conference: pages 147—151, 1988.

> 17 300 citations
(Hiver 2020)

Détecteur de Harris

- Matrice du second moment

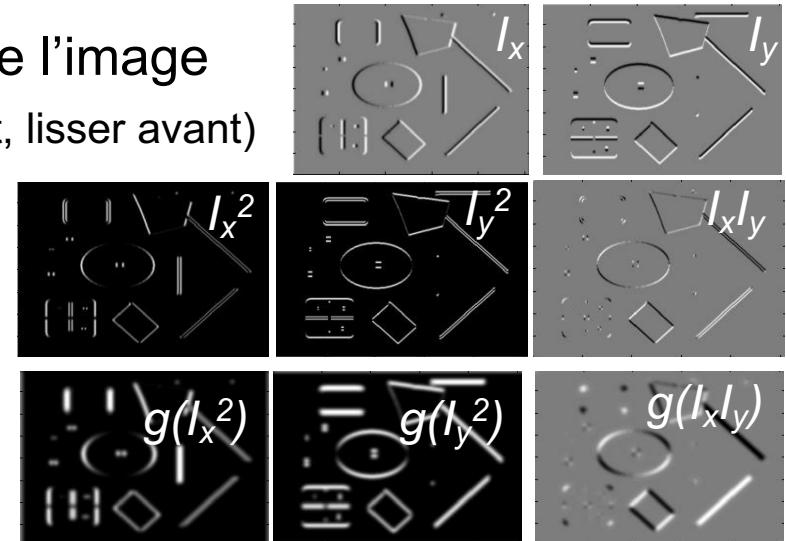
$$\mu(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

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

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

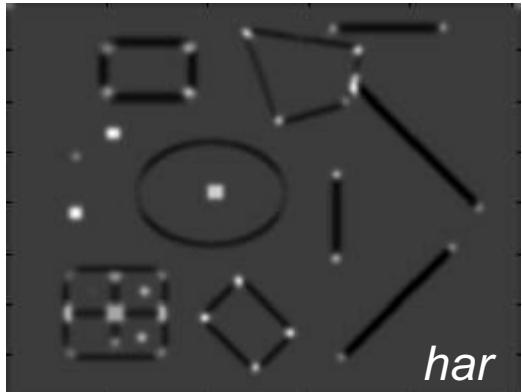
- 3. Filtre gaussien $g(\sigma_l)$

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

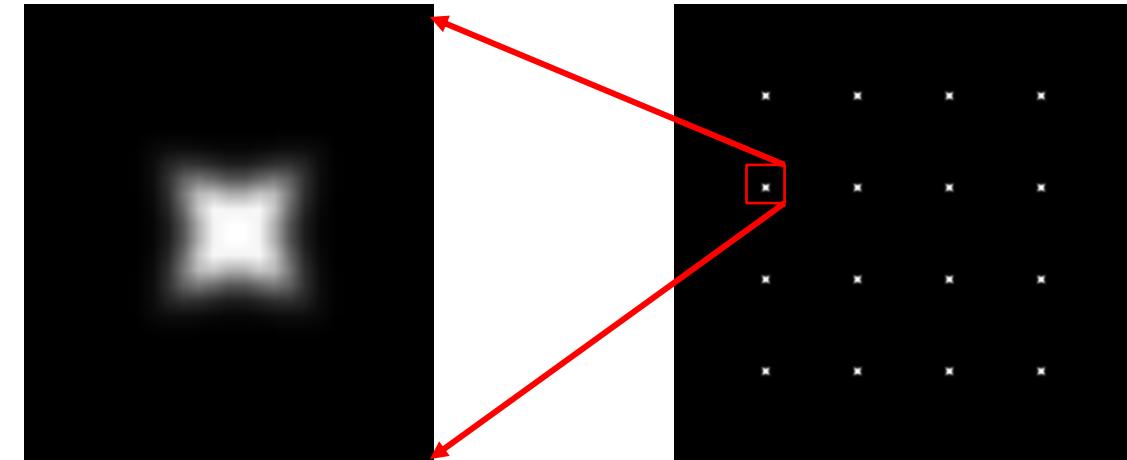


- 4. Fonction de similitude aux coins (*cornerness*)
– les deux valeurs propres sont grandes

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



Comparaison entre Harris et λ_{min}



Harris
operator

λ_{min}

```
>>> from skimage.feature import corner_harris, corner_peaks
>>> square = np.zeros([10, 10])
>>> square[2:8, 2:8] = 1
>>> square.astype(int)
array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 1, 1, 1, 1, 0, 0, 0],
       [0, 0, 1, 1, 1, 1, 1, 0, 0, 0],
       [0, 0, 1, 1, 1, 1, 1, 1, 0, 0],
       [0, 0, 1, 1, 1, 1, 1, 1, 1, 0],
       [0, 0, 1, 1, 1, 1, 1, 1, 1, 0],
       [0, 0, 1, 1, 1, 1, 1, 1, 1, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
>>> corner_peaks(corner_harris(square), min_distance=1)
array([[2, 2],
       [2, 7],
       [7, 2],
       [7, 7]])
```

Source : [Documentation de scikit-image](#)

Exemple : DéTECTeur de Harris

1. Calcul de la réponse du détecteur de Harris
2. Seuillage global
3. Détection des maximums locaux
4. Affichage des coins détectés

Les habits du Dr Horacio Arruda à dessiner
Source : [Bach Illustration & Radio-Canada](#)



Exemple : DéTECTeur de Harris

- 1. Calcul de la réponse du détECTeur de Harris**
2. Seuillage global
3. DéTECTION des maximums locaux
4. Affichage des coins détECTés

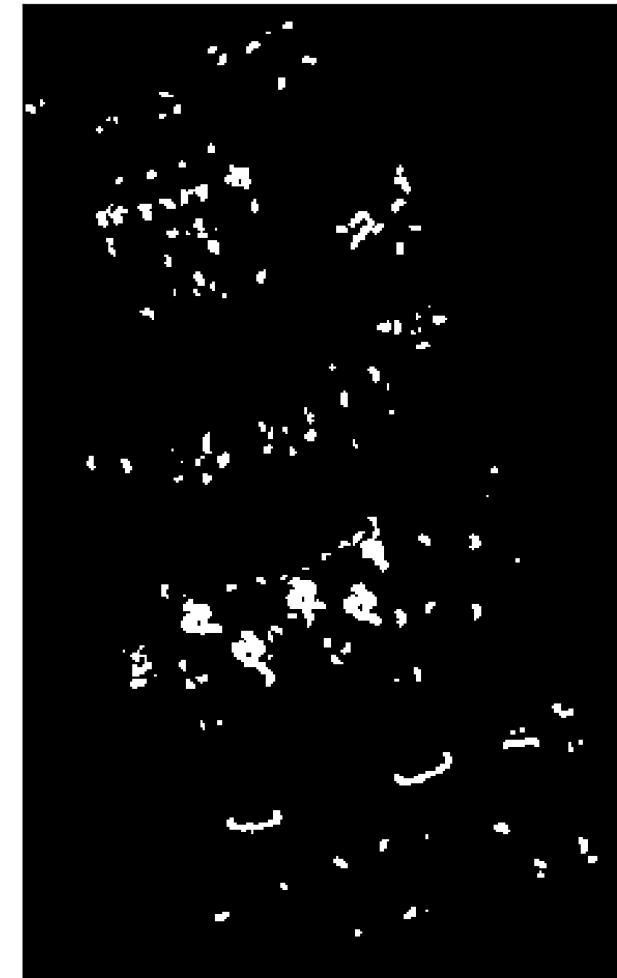
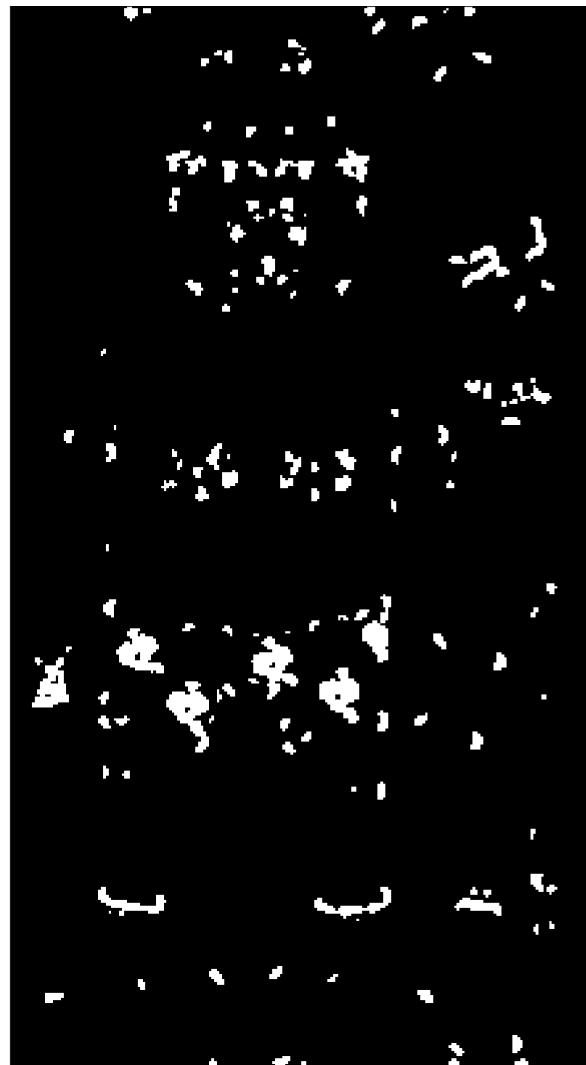
```
# Calcul du détECTeur de Harris (f1, f2)
f1 = skimage.feature.corner_harris(img1_gray)
f2 = skimage.feature.corner_harris(img2_gray)
```



Exemple : DéTECTEUR de Harris

1. Calcul de la réponse du détecteur de Harris
2. **Seuillage global**
3. Détection des maximums locaux
4. Affichage des coins détectés

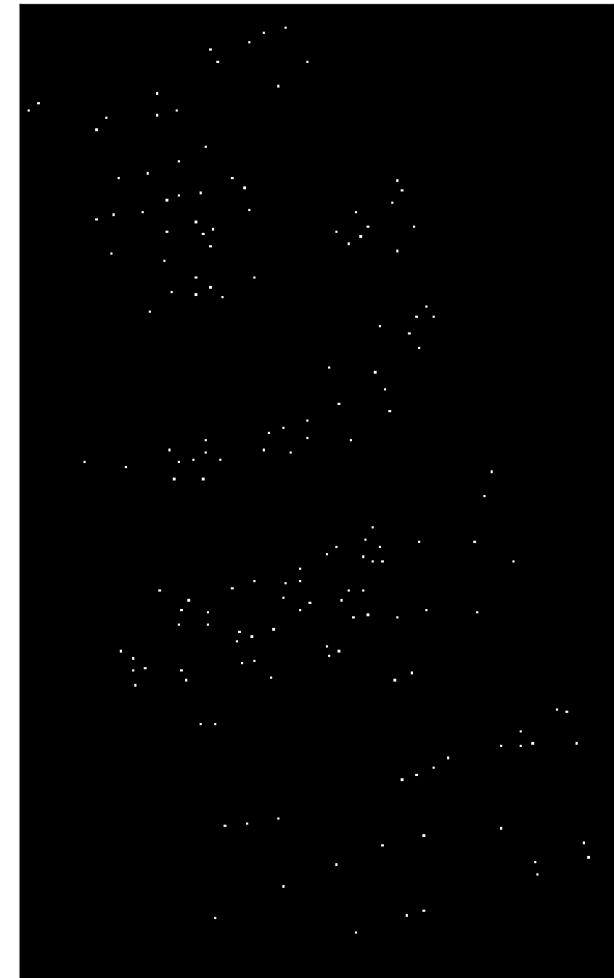
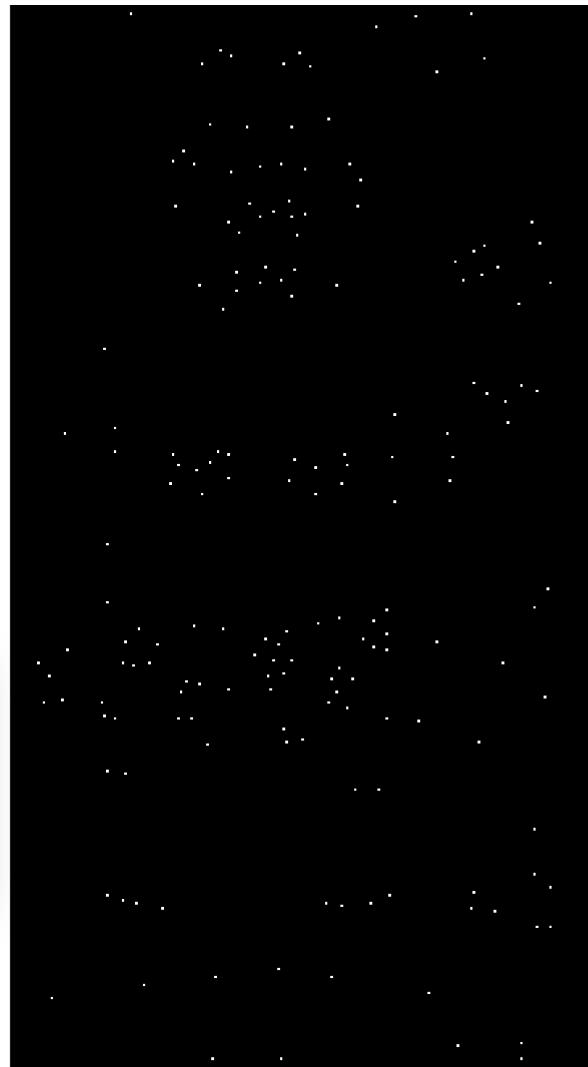
```
# Application d'un seuil
s = 0.5
s1 = f1 > s
s2 = f2 > s
```



Exemple : DéTECTEUR de Harris

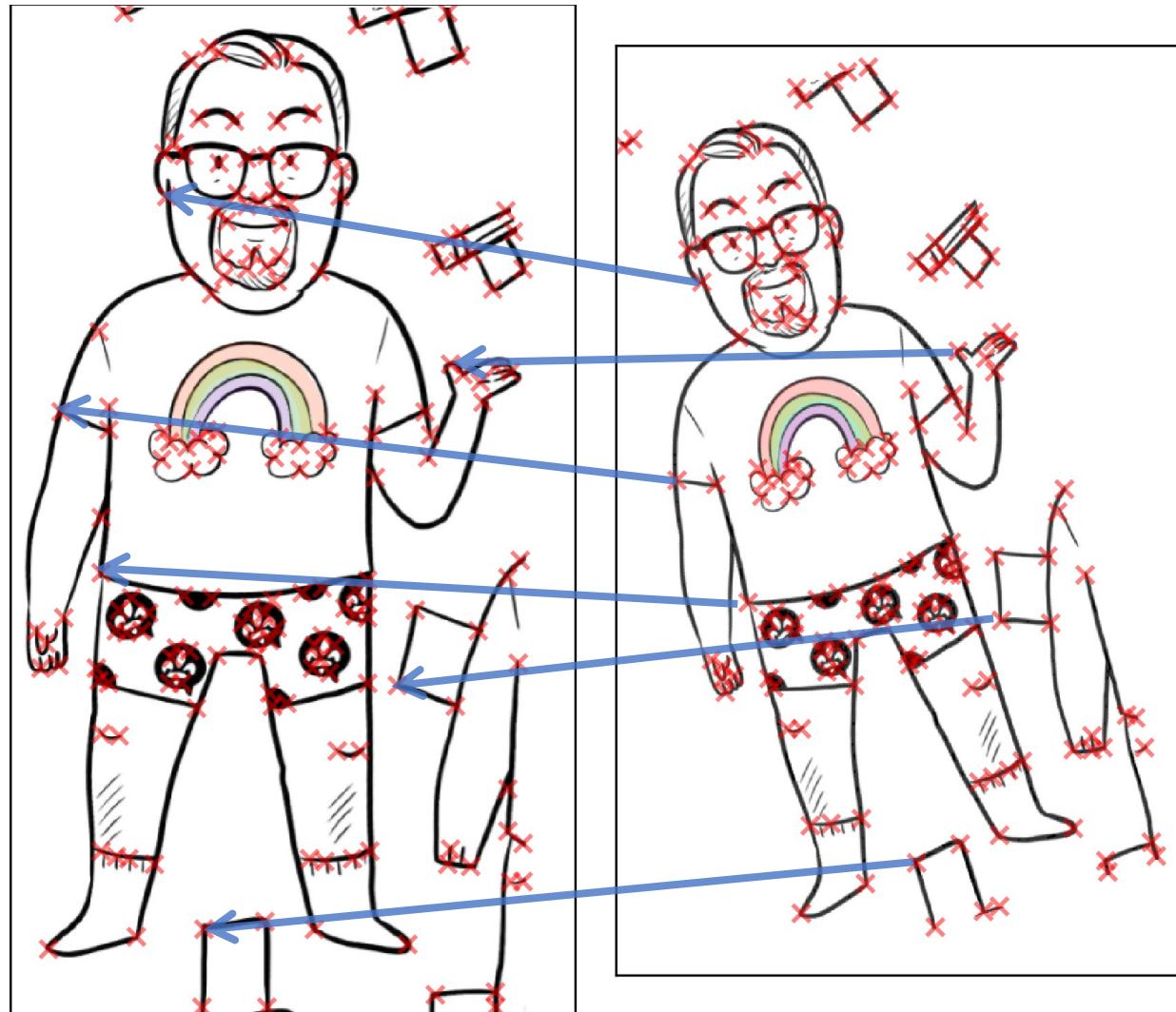
1. Calcul de la réponse du détecteur de Harris
2. Seuillage global
- 3. DéTECTION des maximums locaux**
4. Affichage des coins détectés

```
# Trouver les maximum locaux
min_dist = 3
c1 = skimage.feature.peak_local_max(img1_gray, labels=s1,
                                      min_distance=min_dist)
m1 = np.zeros_like(img1_gray, dtype=bool)
m1[c1[:,0], c1[:,1]] = True
```



Exemple : DéTECTEUR de Harris

1. Calcul de la réponse du détecteur de Harris
2. Seuillage global
3. Détection des maximums locaux
- 4. Affichage des coins détectés**

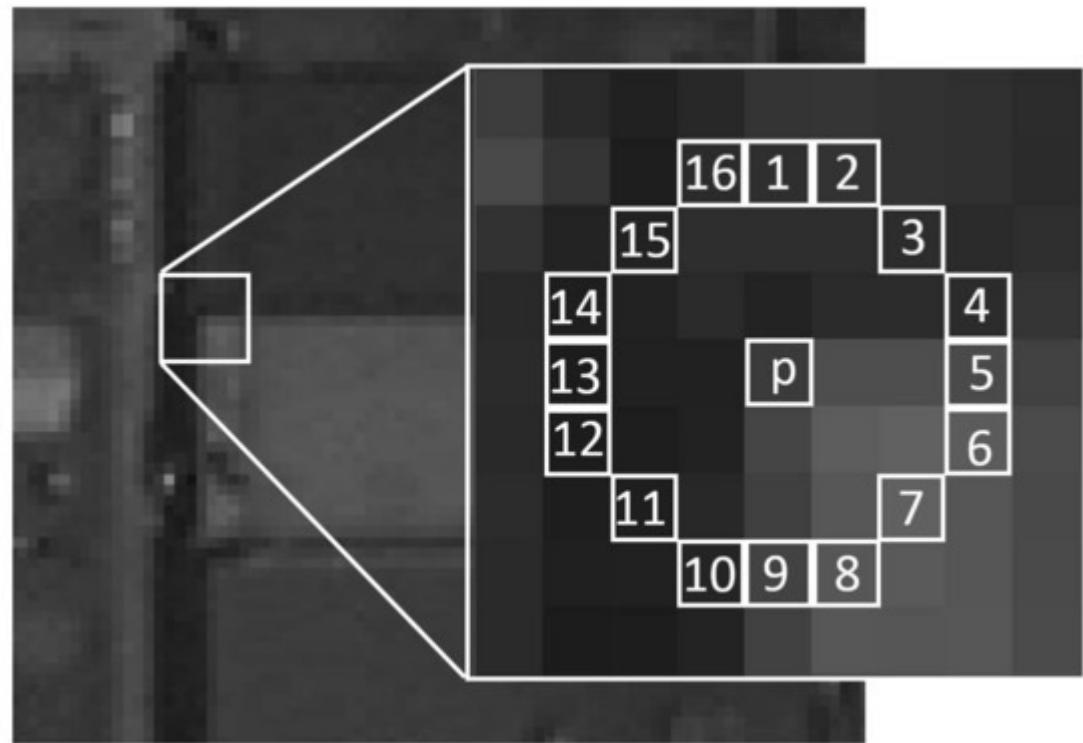


Plusieurs types de détecteurs de coins

<code>skimage.feature.corner_kitchen_rosenfeld</code>(image)	Compute Kitchen and Rosenfeld corner measure response image.
<code>skimage.feature.corner_harris</code>(image[, ...])	Compute Harris corner measure response image.
<code>skimage.feature.corner_shi_tomasi</code>(image[, sigma])	Compute Shi-Tomasi (Kanade-Tomasi) corner measure response image.
<code>skimage.feature.corner_foerstner</code>(image[, sigma])	Compute Foerstner corner measure response image.
<code>skimage.feature.corner_subpix</code>(image, corners)	Determine subpixel position of corners.
<code>skimage.feature.corner_peaks</code>(image[, ...])	Find corners in corner measure response image.
<code>skimage.feature.corner_moravec</code>(image[, ...])	Compute Moravec corner measure response image.
<code>skimage.feature.corner_fast</code>(image[, n, ...])	Extract FAST corners for a given image.
<code>skimage.feature.corner_orientations</code>(image, ...)	Compute the orientation of corners.
<code>skimage.feature.structure_tensor</code>(image[, ...])	Compute structure tensor using sum of squared differences.
<code>skimage.feature.structure_tensor_eigvals</code>(...)	Compute Eigen values of structure tensor.

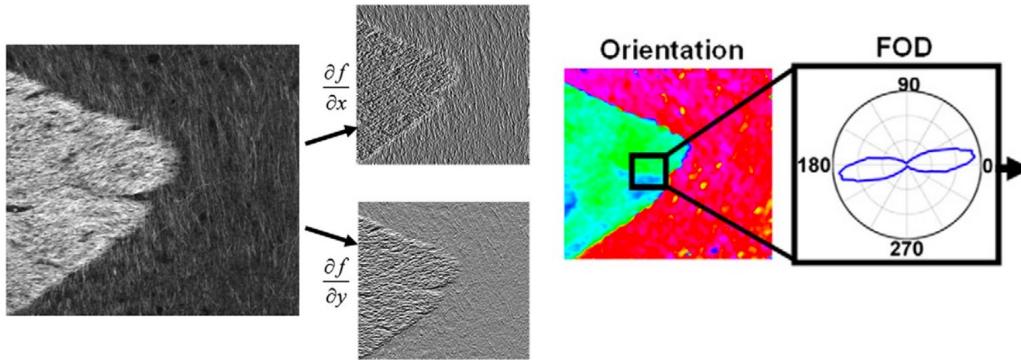
Détecteur de coin FAST

- FAST : *Features from accelerated segment test*
- Computationnellement efficace
- Principes
 - p : pixel considéré pour la détection de coin, ayant une intensité v_p
 - Si N voisins parmi les 16 pixels voisins sur un cercle autour de p sont plus clairs / sombres que p pour un seuil prédéterminé, c'est un coin
 - Répéter pour tous les pixels
- Cette méthode peut être combinée à de l'intelligence machine pour améliorer les performances



Source : [Wikimedia](#)

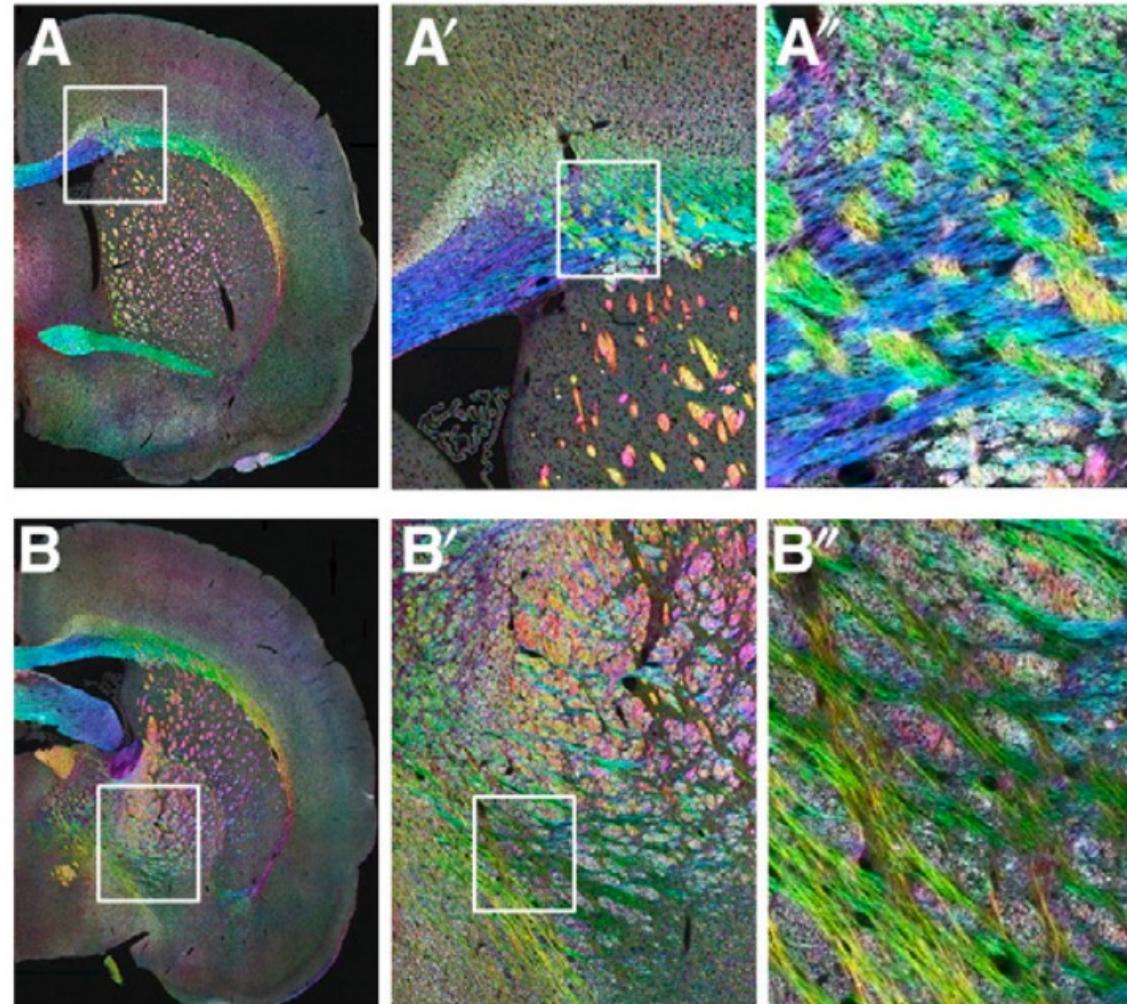
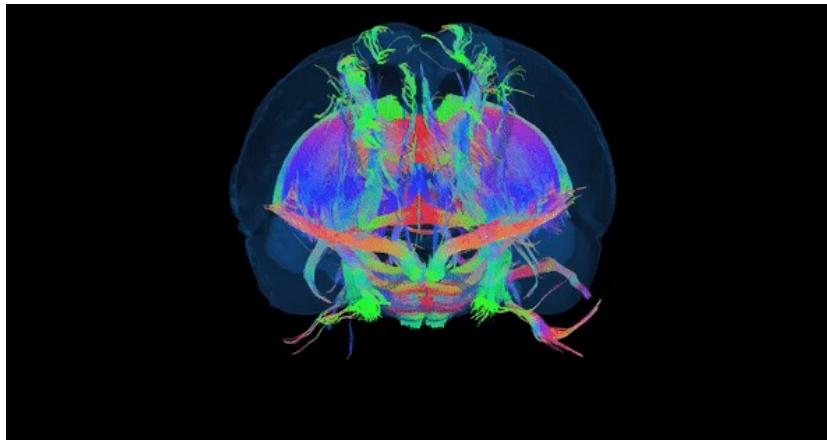
Exemple : Tenseur de structure 2D Détection de l'orientation des neurones



<https://doi.org/10.1016/j.neuroimage.2012.06.042>

Tractographie IRMd
de la matière
blanche dans un
cerveau de rat de 40
jours.

[Duke CIVM](#)



Matrice Hessienne

- Similaire au tenseur de structure, mais utilisant les 2^e dérivées partielles plutôt que les 1^e dérivées partielles

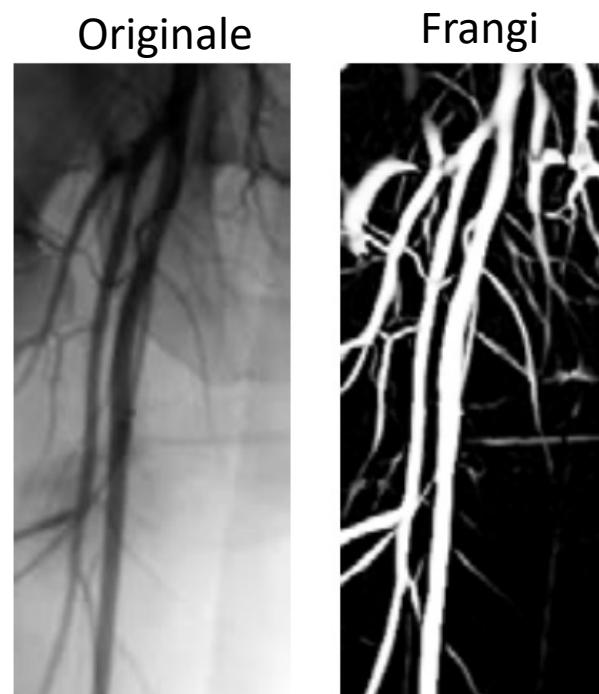
$$H(x, y) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

- Décrit la courbure locale
- Utilisée pour trouver les points d'inflections, détection de blob, calcul d'un indice de forme, détections des tubes, calcul de l'orientation ...

Exemple : Filtre de Frangi

- Détection des contours continus
- Exemple : vaisseaux sanguins, rides, rivières ...
- Basé sur l'analyse des valeurs / vecteurs propres de la matrice Hessienne.
- C'est un **filtre détecteur de forme**

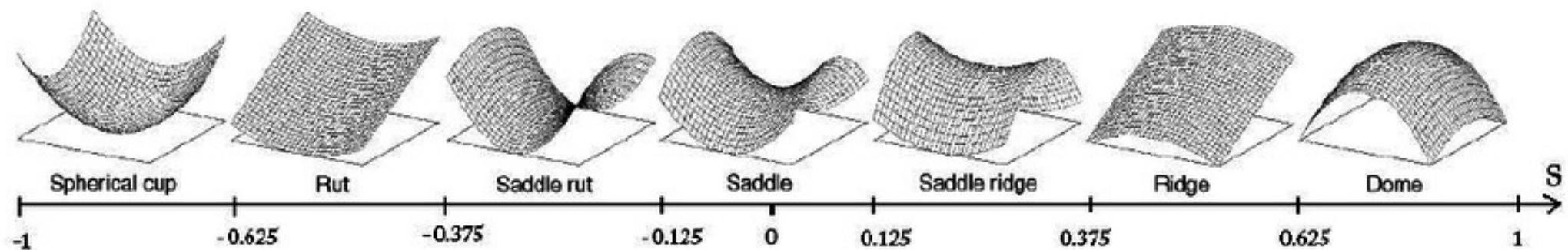
λ_1	λ_2	Patron d'orientation
Bruité et petit	Bruité et petit	Bruit, pas d'orientation
Petit	Grand et négatif	Structure tubulaire claire
Petit	Grand et positif	Structure tubulaire sombre
Grand et négatif	Grand et négatif	Structure blob claire
Grand et positif	Grand et positif	Structure bloc sombre



Augmentation du contraste
des vaisseaux sanguins à
l'aide d'un filtre de Frangi.
Angiographie rayon-x
(Source : [Frangi1998](#))

Exemple : Indice de forme (*Shape index*)

- Mesure de la courbure locale
- Suppose que l'image est une surface 3D, et que l'intensité représente la hauteur de la surface
- Obtenu en analysant les valeurs propres de la Hessienne
- Démo : [scikit-image](#)



Source : [Vezzetti2014](#)

Invariance et Covariance

- Est-ce que les positions des caractéristiques sont *invariantes* face à des transformations photométriques (luminosité, saturation, ...)
- Sont-elles *covariantes* avec les *transformations géométriques* (*translation, rotation, ...*)
- **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.

À suivre ...

