Traitement des images
(Partie 2: segmentation de bas niveau)

Patrick Hébert, Samy Metari & Denis Laurendeau (Dernière révision : juin 2017)

Références utiles:

Sonka et al : 5.3.2 à 5.3.5, 5.3.10, 6.2, 6.2.1

Introduction

Cette partie du cours aborde les éléments suivants :

1. Définition de “**caractéristiques**” (“feature”) d’une image
2. Détection des **arêtes**
3. Détection des **coins** et **points isolés** “stables” et invariants

Les **caractéristiques** d'intérêt dans les images

Définition 1:

Une “**caractéristique**” (“feature”) est une partie ***significative***, ***locale*** et ***détectable*** d’une image

Définition 2:

Les ***parties significatives*** incluent:

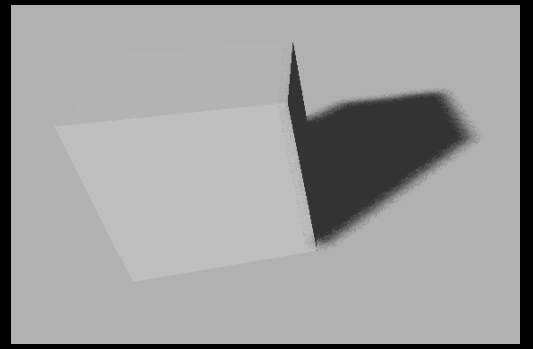
- les ***arêtes*** et les ***contours*** (i.e. discontinituité dans une propriété de l’image comme l’illuminance, la texture, la couleur, l’ombre, l’orientation de surface, etc.)
- les ***coins***
- les ***régions*** (i.e. les zones de l’image qui partagent la même caractéristique)

Exemples

Arêtes d’illuminance

Arêtes de texture

Arêtes de couleurs



Arêtes d’ombres

Détection de coins

Segmentation en régions
de couleurs indentiques

La détection des ***arêtes*** : approches par dérivée première

Les ***étapes classiques*** de détection des pixels d'arêtes sont les suivantes:

1. ***Filtrage*** du bruit avant la dérivation

2. ***Rehaussement*** (i.e. détection) des arêtes par l'application d'une ***dérivée*** sur l'image d'illuminance filtrée

3. ***Localisation*** précise des arêtes (élimination des non-maximums et seuillage)

Les opérations de ***détection*** et de ***localisation*** des arêtes sont antagonistes et il faut trouver un ***compromis*** entre les deux pour obtenir des performances satisfaisantes:

1. une ***détection fiable*** peu affectée par le bruit résulte généralement en une ***localisation imprécise***

2. une ***localisation précise*** des arêtes résulte en une ***détection peu fiable*** et très ***affectée par le bruit*** (i.e. le nombre de pixels d'arêtes associés au bruit est élevé)

- Les arêtes ***d'illuminance*** résident généralement sur le ***contour*** des "objets" dans l'image.

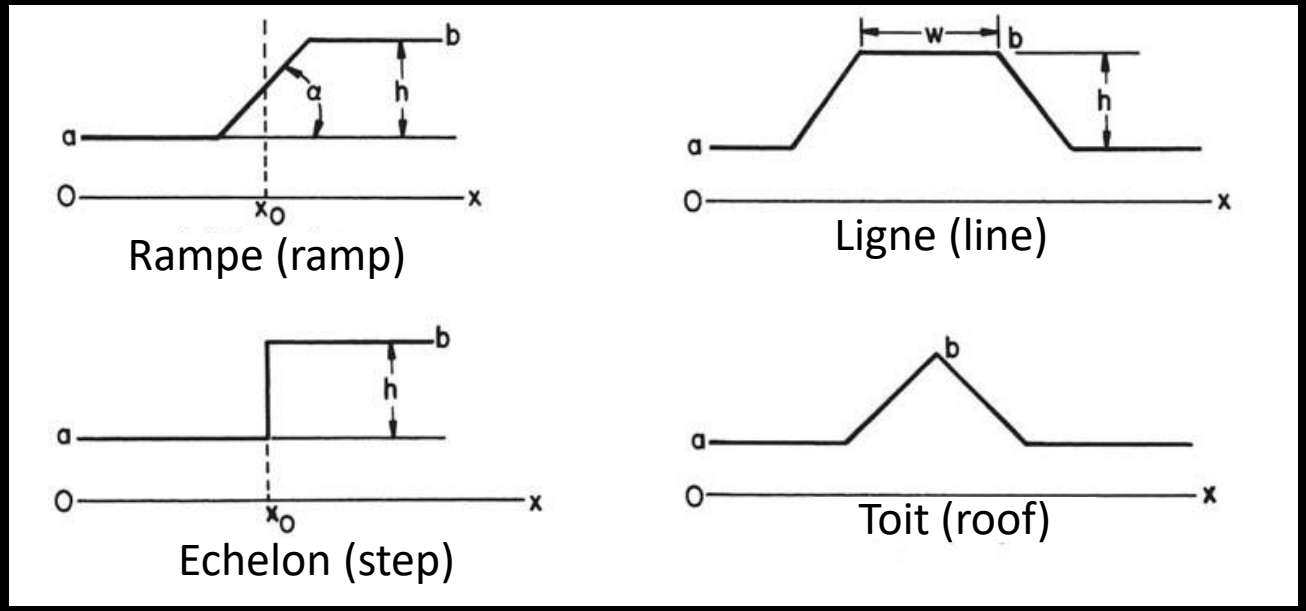
- Les ***contours*** correspondent aux changements d'illuminance importants

- La définition "***d'objet***" n'est pas claire...

- La détection des pixels d'arêtes se fait généralement par l'application d'un opérateur de ***dérivation*** (dérivée première ou seconde) sur l'image

- Le calcul de la dérivée entraîne aussi une augmentation des effets du ***bruit***.

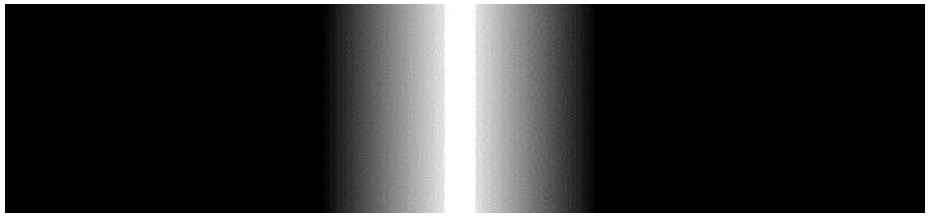
Types d’arêtes



Tiré de Pratt

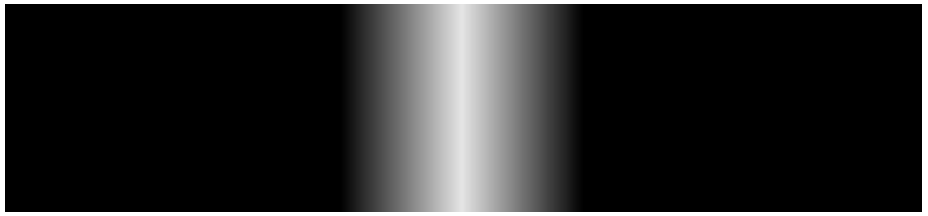
Exemples

Rampe (ramp)



Ligne (line)

Echelon (step)



Toit (roof)

Tiré de Gonzalez et Woods

Rehaussement (détection) des arêtes

La ***détection*** des ***arêtes*** se fait généralement par le calcul du ***gradient*** de l'image d'illuminance E(x,y):

$$\nabla E(x,y) = \left[\frac{\delta E(x,y)}{\delta x}, \frac{\delta E(x,y)}{\delta y}\right] \quad (1)$$

Le gradient est une quantité ***vectorielle*** ayant une ***amplitude*** et une ***orientation***.

En pratique, comme l'image est discrète (E(i,j)), on utilise des opérateurs de gradient discrets (horizontal (colonnes) et vertical (lignes)) convolués avec l'image

Dérivée "horizontale" $$D_x = [-1 \quad 0 \quad 1] \tag{2}$$

Dérivée "verticale" $$D_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \tag{3}$$

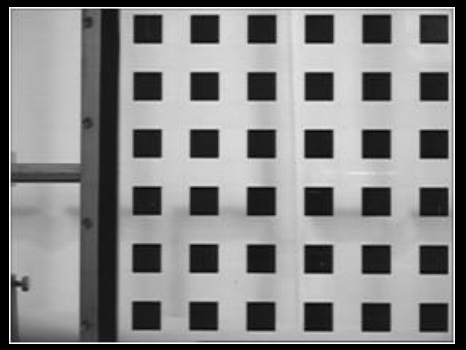
Comme la dérivée est sensible au bruit, il est recommandé de passer un filtre gaussien avant la dérivée

On peut tirer profit du fait que la ***convolution*** est une opération ***linéaire*** (que ce soit pour le filtrage gaussien ou pour la dérivation) pour écrire que:

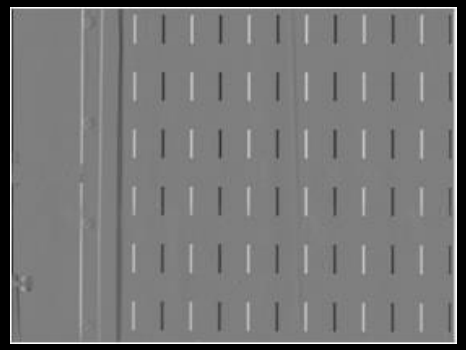
$$D * (G * E) = (D * G) * E \tag{4}$$

On peut donc ***filtrer*** l'image et ***appliquer*** la dérivée (gauche de (4) ou ***appliquer*** la dérivée au noyau du filtre et ***filtrer*** l'image avec le noyau résultant (droite de (4)).

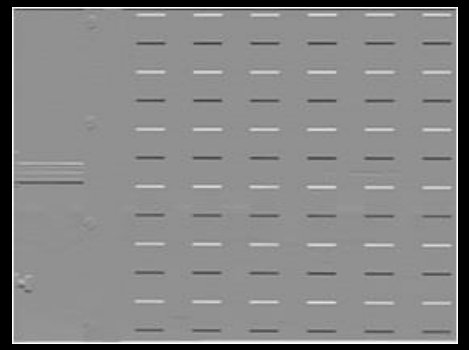
Exemple



$G * E$



$E_x = D_x * (G * E)$

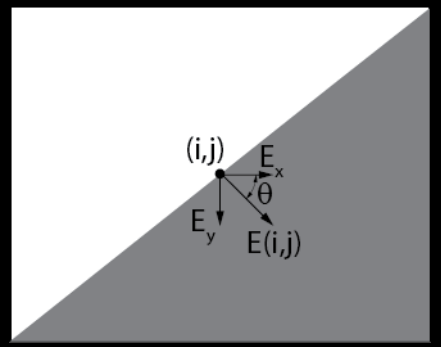


$E_y = D_y * (G * E)$

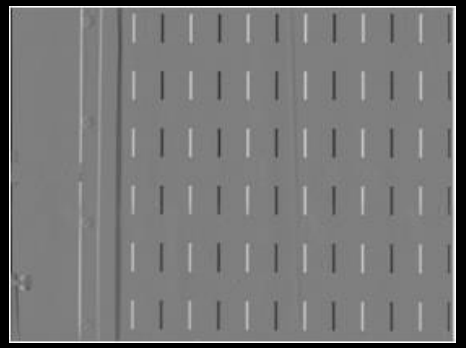
Une fois E_x et E_y calculés, on s'intéresse au ***module*** et à ***l'orientation*** du gradient à chaque pixel:

Module $$E_s(i,j) = \|\nabla E(i,j)\| = \sqrt{E_x^2 + E_y^2} \quad (5)$$

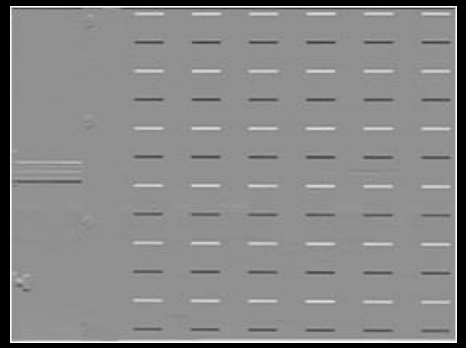
Orientation $$E_\theta = \arctan\left[\frac{E_y}{E_x}\right] \quad (6)$$



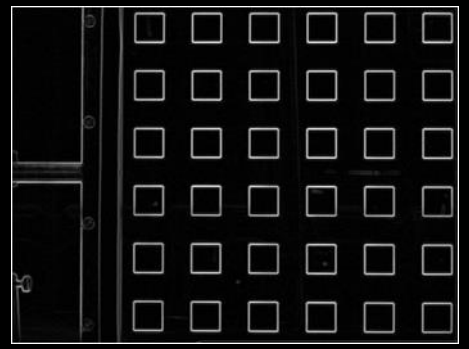
Exemple avec l’échiquier



$$E_x = D_x * (G * E)$$



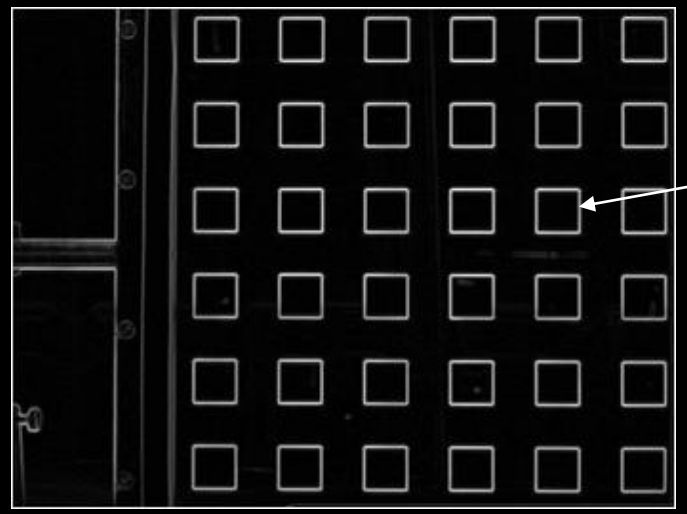
$$E_y = D_y * (G * E)$$



$$E_s = \sqrt{E_x^2 + E_y^2}$$

Amplitude du gradient

Pour détecter les pixels d’arête, il suffit d’appliquer un seuil sur l’amplitude du gradient et de ne conserver que les pixels au-dessus de ce seuil. Le seuil peut être choisi automatiquement en construisant un histogramme de l’amplitude du gradient.



pixels dont l’amplitude du gradient dépasse un seuil

La détection des ***arêtes*** : approches par dérivée seconde

Il est aussi possible de détecter les pixels d'arête en calculant la ***dérivée seconde*** de l'image.

On parle dans ce cas de l'opérateur ***Laplacien***.

$$\nabla^2 E(x,y) = \frac{\partial^2 E(x,y)}{\partial x^2} + \frac{\partial^2 E(x,y)}{\partial y^2} \tag{7}$$

Le résultat est une quantité ***scalaire***.

Le Laplacien étant une ***dérivée seconde***, son calcul est encore plus affecté par le ***bruit*** que le gradient.

Il est donc important de ***filtrer*** l'image avec un filtre ***gaussien*** avant d'appliquer la dérivée seconde.

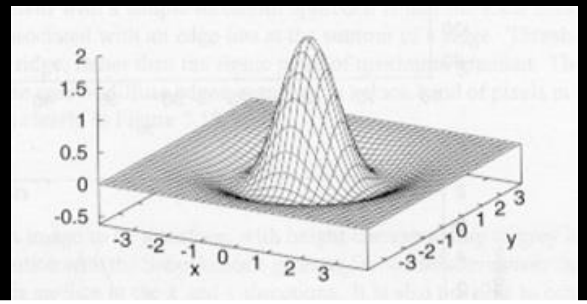
Comme dans le cas du gradient, on peut tirer profit du fait que le filtrage gaussien et le Laplacien sont des opérations linéaires. On peut écrire:

$$G * (\nabla^2 * E) = (G * \nabla^2) * E \qquad (8)$$

En tirant profit de l’associativité, on a que:

$$\nabla^2 * (G * E) = (\nabla^2 * G) * E$$

Laplacien d’une Gaussienne (Opérateur LoG)



Laplacien d’une gaussienne inversé

Version discrète 3 x 3

$$L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad (9)$$

Tiré de Efford

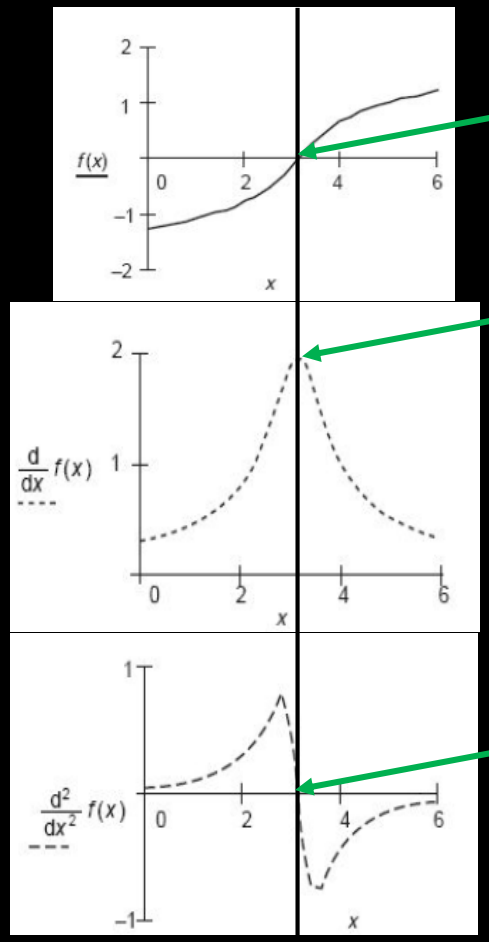
Propriétés importantes du Laplacien de Gaussienne LoG) :

1. Opérateur à ***symétrie de révolution*** (i.e. ne fournit aucune information sur l'orientation de l'arête)

2. Les pixels d'arête sont situés au ***passage par zéro du LoG***

3. Produit des contours ***fermés***

Illustration de la détection des pixels d'arête avec l'opérateur LoG



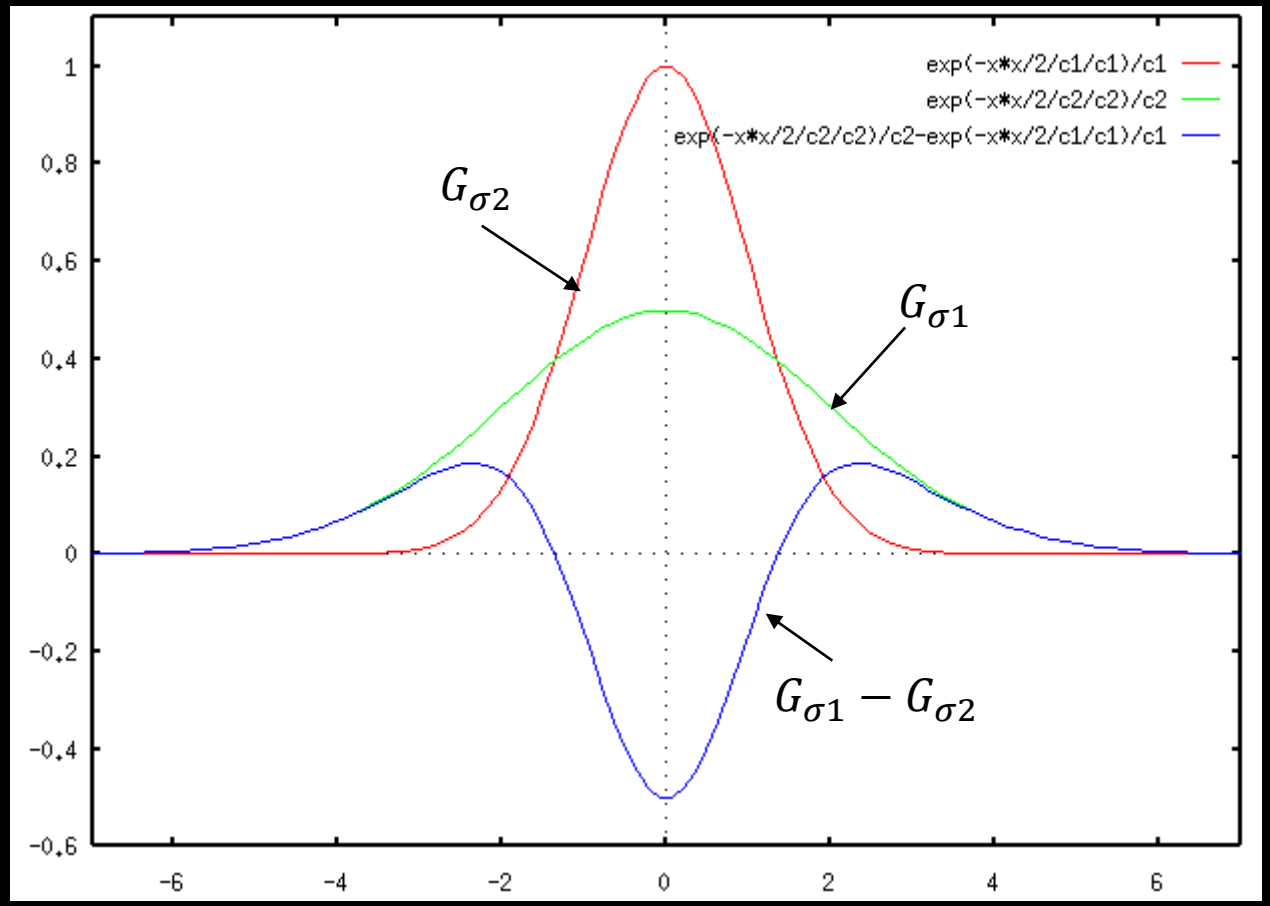
pixel d'arête

l'arête est située au maximum du gradient...

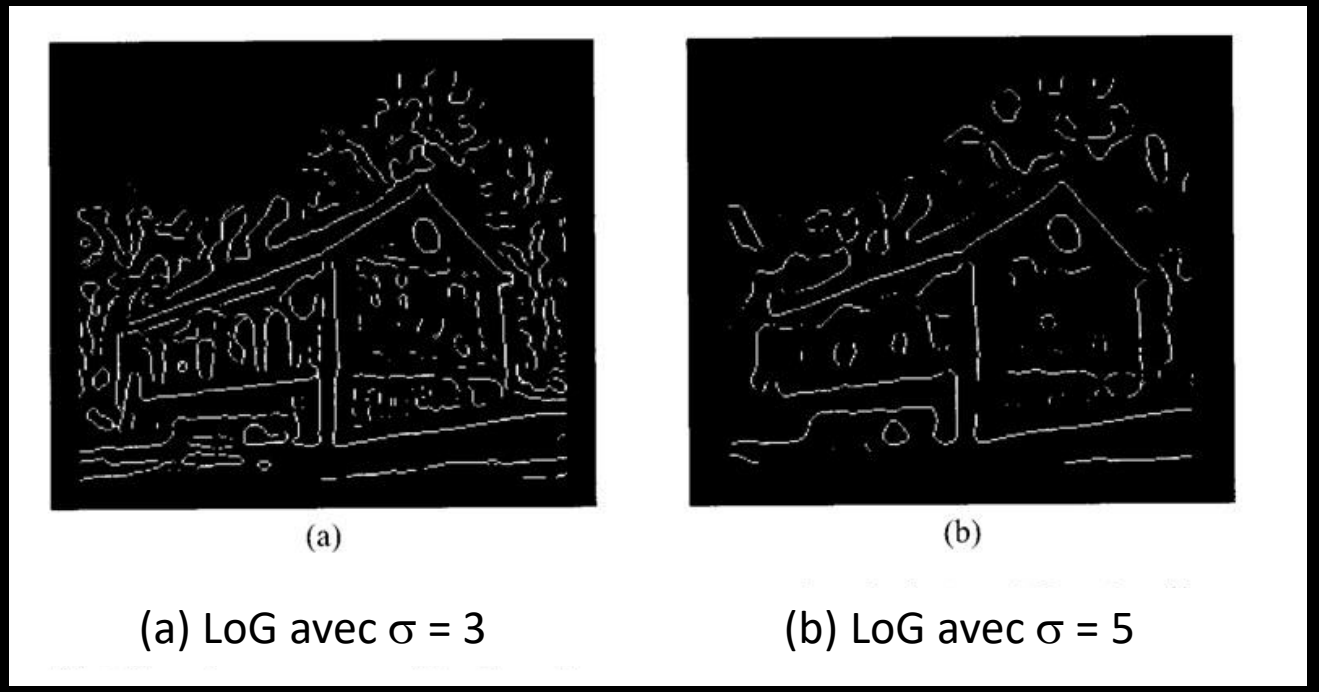
ou au passage par zéro du LoG

Tiré de Nixon

On peut approximer le LoG en prenant la différence de deux gaussiennes (DoG) de valeurs d'écart-type σ différentes



Exemple de résultats



(a) LoG avec $\sigma = 3$ (b) LoG avec $\sigma = 5$

Tiré de Efford

Pour des raisons non-évidentes et non reliées à la discrétisation de l'opérateur, le LoG cause l'apparition d'artéfacts indésirables comme des boucles le long de certaines arêtes et des petites bosses près des coins

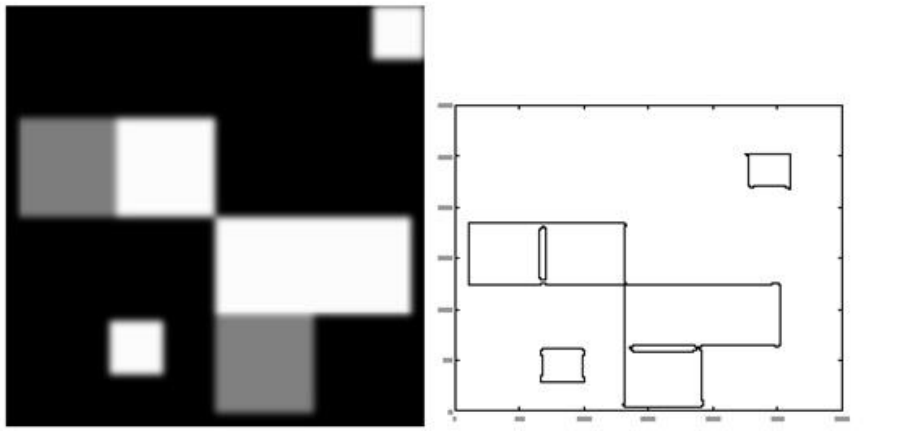


Figure 8.20. Zero crossings of Laplacian of Gaussian output can behave strangely at corners. Firstly, at a right angled corner, the zero crossing bulges out at the corner (but passes through the vertex). This effect is not due to digitisation or to quantization, but can be shown to occur in the continuous case as well. At corners where three or more edges meet, contours behave strangely, with the details depending on the structure of the contour marking algorithm — this algorithm (the one shipped with Matlab) produces curious loops. This effect can be mitigated with careful design of the contour marking process, which needs to incorporate a fairly detailed vertex model.

Tiré de Forsyth

La détection des ***arêtes*** par dérivée première : quelques autres opérateurs discrets

Opérateur de ***Roberts*** (peu utilisé, mais célèbre sur le plan historique)

$$R_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad R_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{10}$$

Opérateur de ***Prewitt*** (très utilisé à cause de sa simplicité)

$$P_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad P_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{11}$$

Opérateur de ***Sobel*** (donne plus d'importance aux pixels sur le 4-voisins du pixel central du masque de convolution)

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{12}$$

La détection des ***arêtes*** par dérivée première : opérateur optimal de ***Canny***

L’un des opérateurs de détection d’arêtes les plus célèbres est ***l’opérateur de Canny (Canny edge détector)***

Canny formule le problème de détection d’arêtes comme suit:

1. formulation mathématique d’un modèle d’arête perturbé par du ***bruit blanc gaussien additif***

2. introduction de ***critères de performance*** pour la détection des arêtes

3. proposition d’un ***filtre linéaire***

4. proposition d’une méthode de ***détection*** des arêtes suite à l’application du filtre linéaire

Critères de performance définis par Canny :

1. ***détection*** des pixels d'arête : le but est d'obtenir ***peu*** de ***fausses alarmes*** et ***peu*** de ***fausses détections***

2. ***localisation*** : le but est de ***positionner*** l'arête le plus ***près*** possible de sa position ***réelle*** dans l'image

3. ***réponse*** de l'opérateur : la réponse de l'opérateur doit être ***simple*** dans le ***voisinage*** de l'arête

→ Il faut faire un ***compromis*** entre ***détection*** et ***localisation***

→ Bonne solution : opérateur ***dérivée de gaussienne*** d'une ***taille*** d'au moins ***7 x 7 à 9 x 9 pixels***

Détecteur de Canny – Etape 1

Calculer ***l'amplitude*** E_s du gradient et son ***orientation*** E_θ en chaque pixel de l'image

Cette tâche peut être faite de deux façons :

Convolution avec des opérateurs directionnels (0°, 45°, 90°, 135°)

$$E_{s-0^\circ} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad E_{s-45^\circ} = \begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix}$$

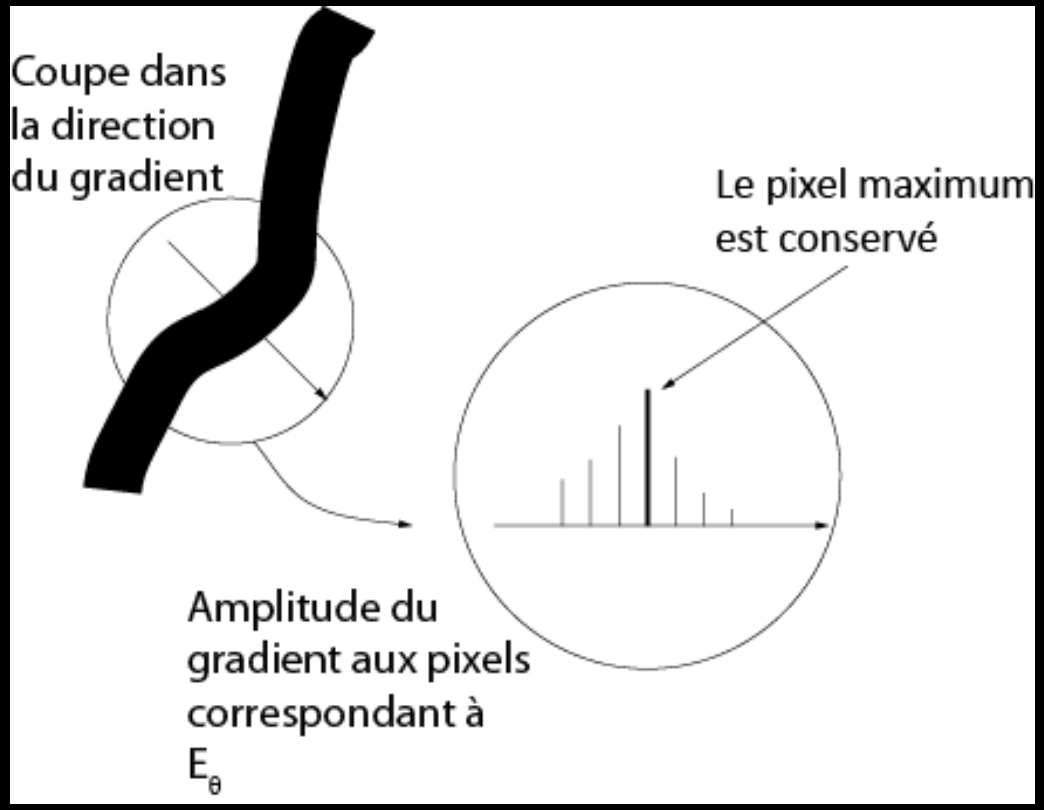
$$E_{s-90^\circ} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad E_{s-135^\circ} = \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Filtrer l'image avec une gaussienne et estimer le gradient

Détecteur de Canny – Etape 2

Élimination des non-maximums:

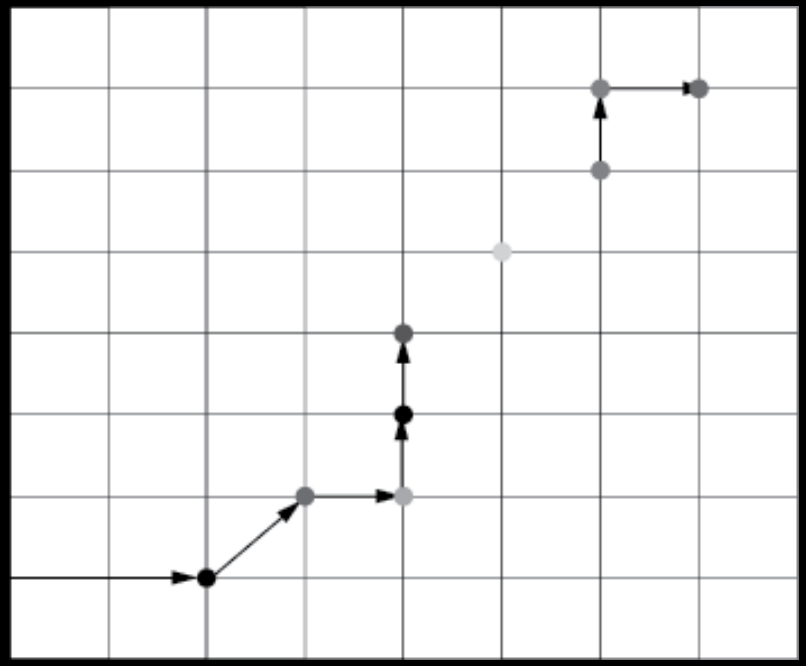
- Seuls les pixels correspondant à ***extremums locaux*** sont considérés comme appartenant aux arêtes et sont conservés pour l'étape suivante
- Un ***extremum local*** correspond à un pixel où ***l'amplitude*** du gradient est ***maximum*** dans un ***voisinage*** correspondant à la ***direction*** du gradient.
- Cette étape conduit à des contours partiels



Détecteur de Canny – Etape 3

Seuillage d'amplitude

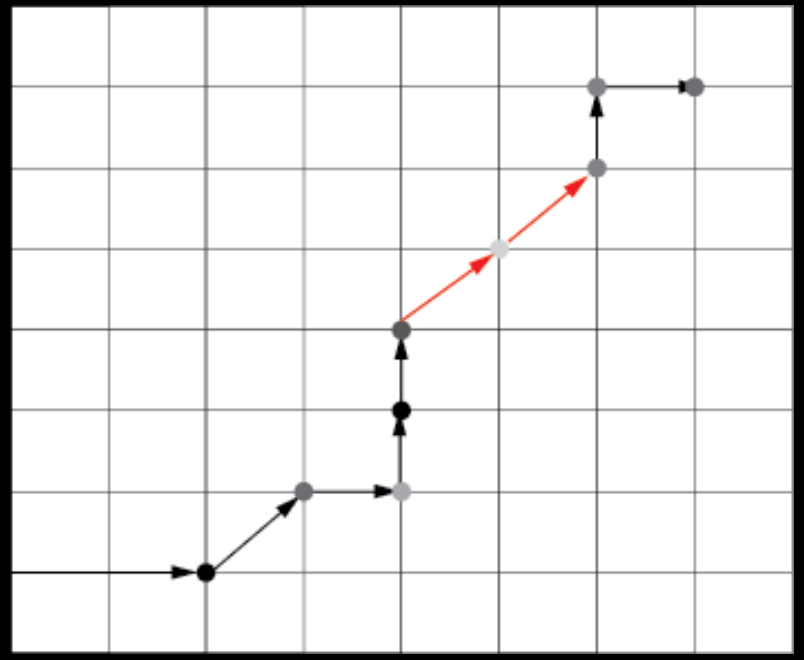
- parmi les pixels ayant franchi l'étape 2, on ne garde que ceux qui "sont significatifs"
- comme l'amplitude du gradient varie beaucoup, le choix d'un seul seuil est difficile
- on procède donc à un seuillage par hystérésis: un seuil haut et un seuil plus bas
- on suit d'abord les pixels dont l'amplitude dépasse le seuil haut

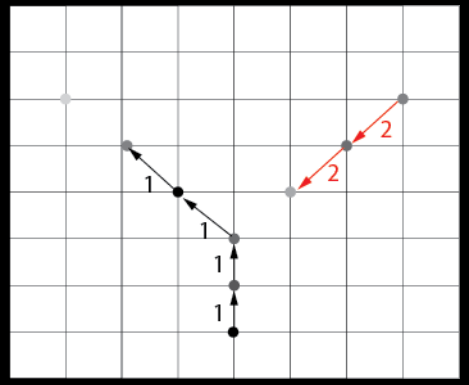


Détecteur de Canny – Etape 3

Seuillage d'amplitude

- on continue ensuite le ***suivi*** en connectant les pixels dont l'amplitude dépasse le seuil haut avec ceux dont l'amplitude dépasse le seuil ***bas***
- les ***jonctions en Y***, qui posent souvent problème dans le suivi de contours, sont ***éliminées à l'étape 2***



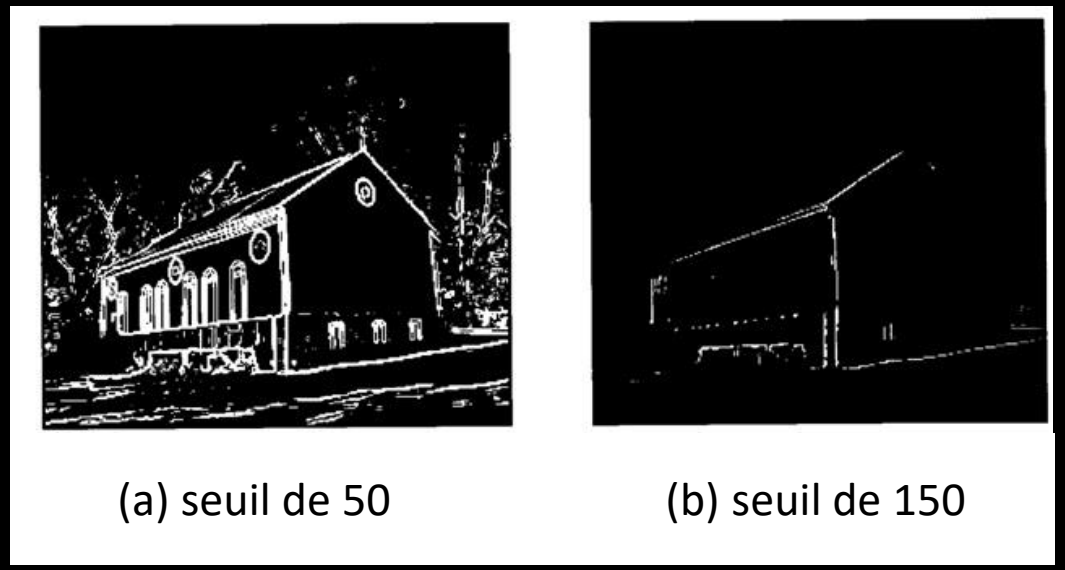


Exemple et analyse

En pratique:

- il y a des pixels d'arête ("edgels") superflus causés par le bruit
- il y a des pixels d'arête non détectés à cause du choix arbitraire des seuils
- il y a des erreurs dans la position et l'orientation des edgels (notamment à cause de la discrétisation)
- pour compenser ces faiblesses,il faut concevoir des algorithmes de suivi d'arêtes de haut niveau (raisonnement à un niveau supérieur au pixel)

Tiré de Efford



(a) seuil de 50 (b) seuil de 150

En complément

Plusieurs améliorations ont été apportées à l'approche de Canny et aux approches de détection d'arêtes en général:

- positionnement sub-pixel des pixels d'arêtes
- accélération du traitement pour les applications en temps réel
- traitement multi-échelles
- traitement spécifique pour les jonctions entre arêtes (jonctions en "L", jonctions en "T", jonctions en "Y"

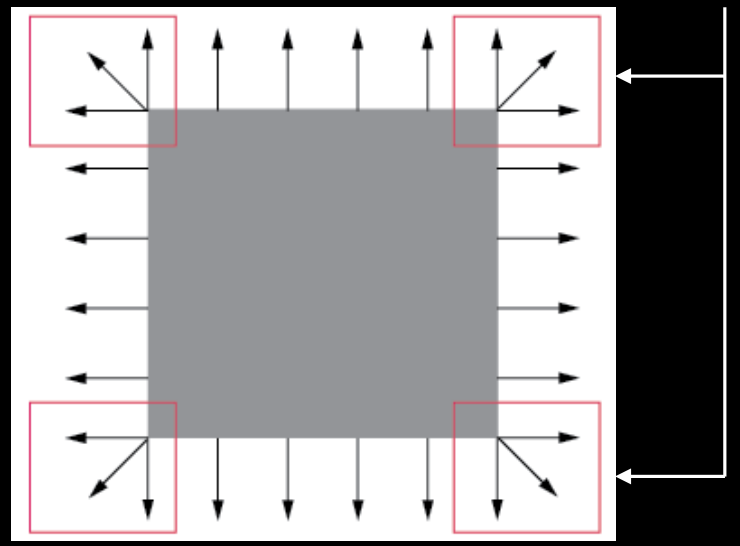
Détection des **coins** et **points isolés** “stables” et invariants

Constat sur les pixels d'arête: les détecteurs se comportent moins bien près des ***coins*** car le ***gradient*** est ***mal défini*** (ou ***discontinu***) en ces endroits de l'image.

Moyen de résoudre ce problème: détecter ***pixels*** où le ***gradient*** est ***fort*** dans ***plus d'une direction –*** ce sont les pixels situés sur les ***coins***

Exemples d'opérateurs pour la détection des coins:

- détecteur de ***Harris***
- détecteur de ***Förstner***



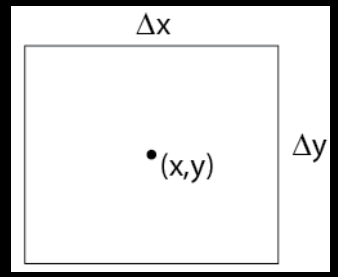
Détection de **coins de Harris**

Etapes de la méthode de Harris

Etape 1: Calculer les images de gradient E_x et E_y

Etape 2: pour tous les points de l'image de gradient, calculer la matrice de covariance du gradient (dans une fenêtre 2N+1 x 2N+1) et en extraire les valeurs propres λ_1 et λ_2 (avec $\lambda_2 \leq \lambda_1$)

Pour visualiser cette étape, on peut prendre un pixel (x,y) et un voisinage (Δx,Δy)



La fonction ***d'autocorrélation*** de l'image en (x,y) est donnée par:

$$C(x,y) = \sum_{w} [I(x_i, y_i) - I(x_i + \Delta x, y_i + \Delta y)]^2 \qquad (13)$$

$$(x_i, y_i) \quad sont\ les\ pixels\ dans\ w$$

Si on approxime $I(x_i + \Delta x, y_i + \Delta y)$ par sa série de Taylor, en se limitant aux 2 premiers termes, on obtient:

$$I(x_i + \Delta x, y_i + \Delta y) = I(x_i, y_i) + \begin{bmatrix} I_x(x_i, y_i) & I_y(x_i, y_i) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \qquad (14)$$

En replaçant (14) dans (13), on a:

$$C(x,y) = \sum_{w} \left[I(x_i, y_i) - I(x_i, y_i) - \begin{bmatrix} I_x(x_i, y_i) & I_y(x_i, y_i) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \right]^2 \quad (15)$$

qu'on peut simplifier comme:

$$C(x,y) = \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} \begin{bmatrix} \sum_{w} I_x^2 & \sum_{w} I_x I_y \\ \sum_{w} I_y I_x & \sum_{w} I_y^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \quad (16)$$

La matrice:

$$\begin{bmatrix} \sum_w I_x^2 & \sum_w I_x I_y \\ \sum_w I_y I_x & \sum_w I_y^2 \end{bmatrix} \quad (17)$$

illustre la structure du gradient de l'image dans un voisinage du pixel (x,y).

Ses valeurs propres λ_1 et λ_2 décrivent l'étalement des valeurs du gradient dans deux directions orthogonales (et peuvent être associées aux "courbures" principales de la fonction d'autocorrélation).

On calcule la valeur R (la réponse de l'opérateur) définie comme:

$$R = det(C(x,y)) - k(trace(C(x,y))^2 \quad (18)$$

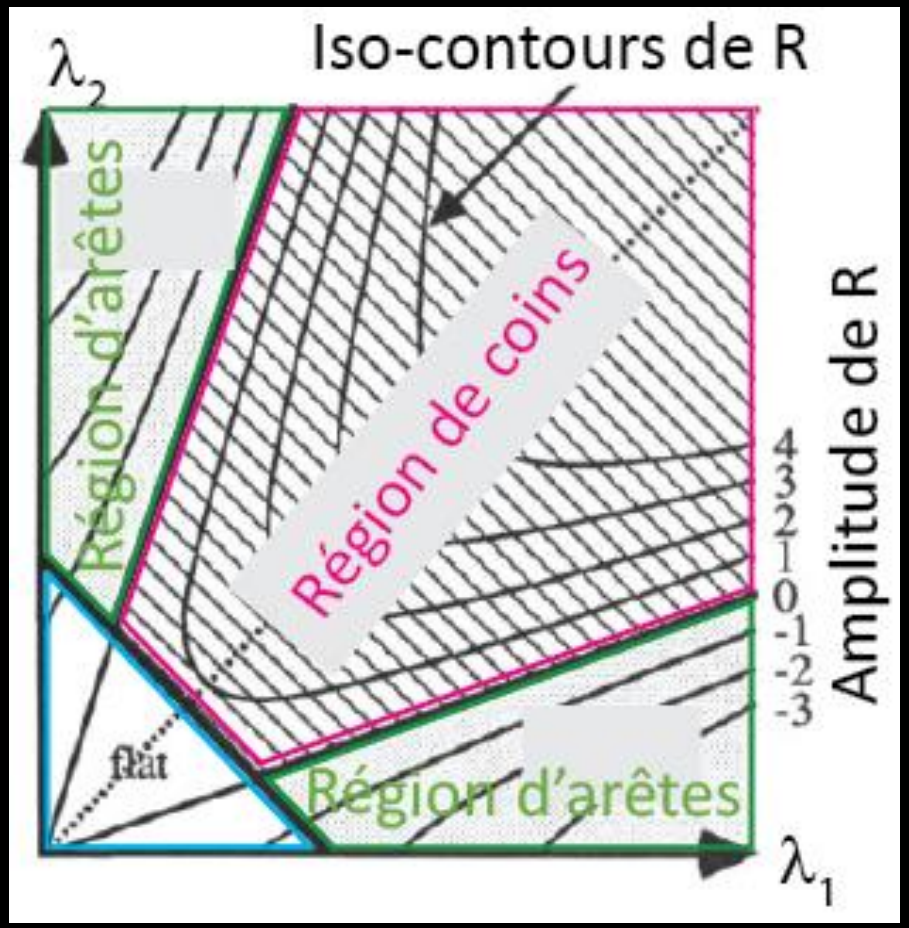
où det(C(x,y)) est le déterminant de C(x,y) qui est égal à $\lambda_1\lambda_2$

et trace(C(x,y)) est la trace de C(x,y) qui vaut $\lambda_1+\lambda_2$. On a donc pour (18) (en assumant que $\lambda_1>\lambda_2$):

$$R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 \quad (19)$$

avec $0.4 \leq k \leq 0.6$

L’analyse des iso-contours de R fournit l’interprétation suivante.

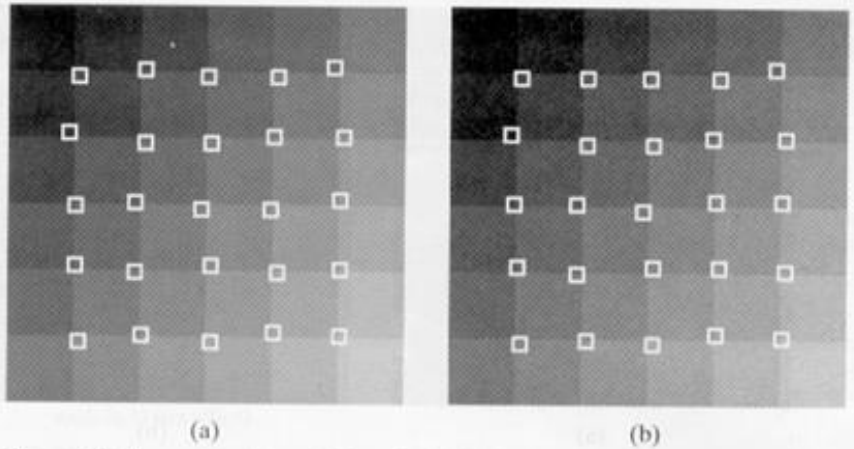


Tiré de “A COMBINED CORNER AND EDGE DETECTOR”, Chris Harris & Mike Stephens, Plessey Research, 1988

Pour repérer les coins significatifs dans l'image, on procède comme suit:

1. on repère les pixels où R est grand
2. on observe les pixels voisins et on élimine ceux pour lesquels R est plus faible
3. après ce tri, on ne conserve que les pixels étant des maximums locaux de R

Résultats



(a) (b)

Figure 4.8 Corners found in a 8-bit, synthetic checkerboard image, corrupted by two realizations of synthetic Gaussian noise of standard deviation 2. The corner is the bottom right point of each 15 × 15 neighbourhood (highlighted).

(a) (b) (c)

Figure 4.9 (a): original image of a building. (b): the 15 × 15 pixel neighbourhoods of some of the image points for which $\lambda_2 > 20$. (c): histogram of λ_2 values across the image.

Tiré de Trucco

Détection de **points isolés stables**

Détecteur **SIFT**

(Scale-Invariant Feature Transform)

Objectif de l'algorithme SIFT

Détecter et **identifier** les **éléments similaires** entre des images numériques avec une **invariance**:

- à **l'échelle;**
- au **cadrage;**
- à **l'angle d'observation** et
- à la **luminosité.**

Cette détection peut avoir plusieurs utilités en vision artificielle:

- **recherche** d'images dans des **bases de données** d'images;
- **suivi** (tracking) d'objets dans une **séquence vidéo;**
- **appariement stéréoscopique** pour la reconstruction 3D (prochain chapitre);
- etc.

Exemple

Dessin

Point “invariant” stable

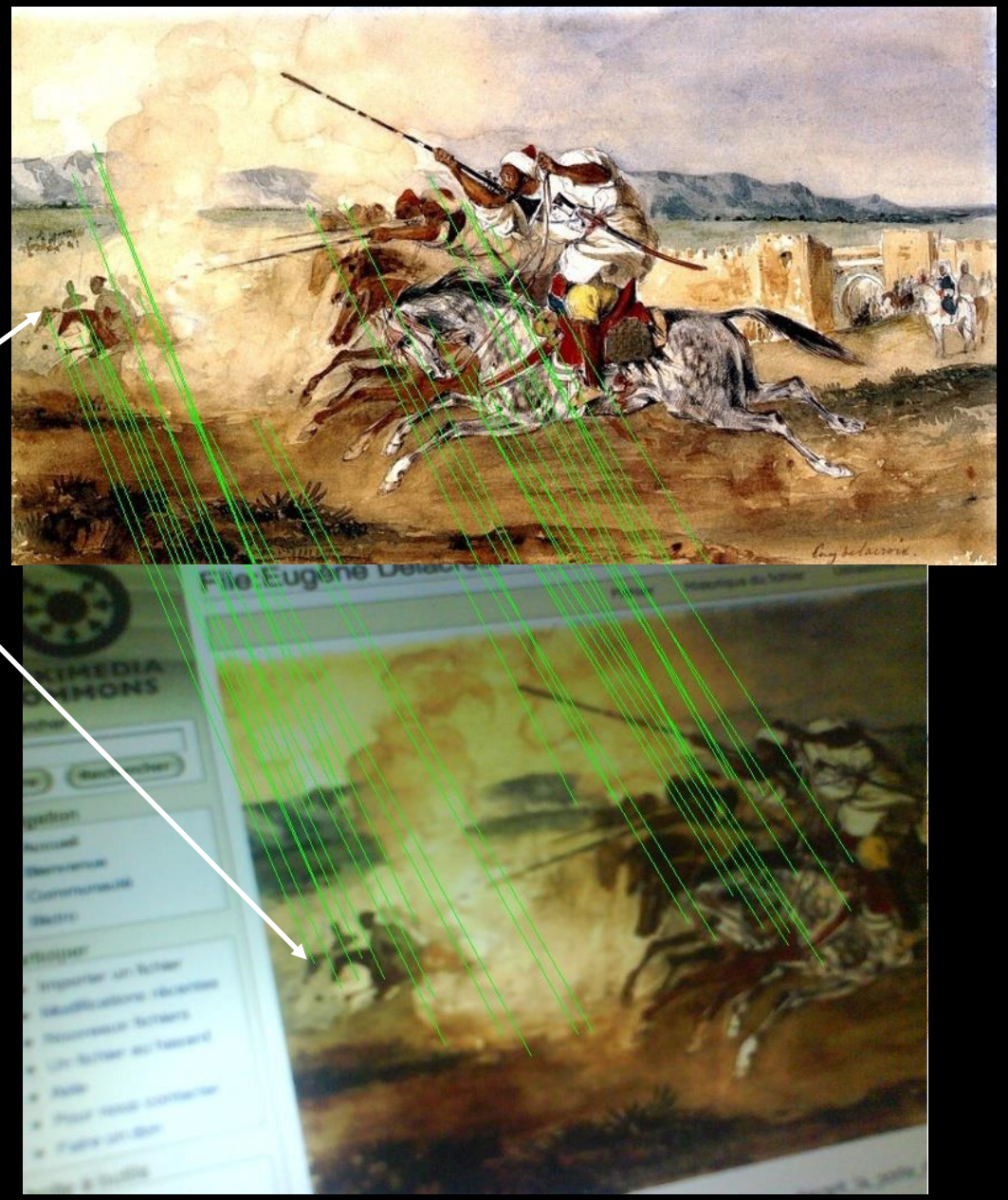


Image du dessin observé sur un écran d’ordinateur avec une orientation, une luminosité et une échelle différentes de celles du du dessin original

La stratégie consiste à trouver un **descripteur** (le descripteur SIFT) pour des **pixels** qui soit **stable** en fonction de l'échelle de l'image

Ce **descripteur** doit être **robuste** aux changements d'orientation, à la luminosité et au cadrage

Seuls les points ayant un descripteur stable sont conservés.

Cette approche a été proposée par David Lowe (professeur à UBC) dans l'article suivant:

David G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", International Journal of Computer Vision, November 2004, Volume 60, Issue 2, pp 91–110

Cette méthode a connu plusieurs variantes et améliorations. Elle demeure un fondement de la vision artificielle.

Les étapes de la méthode SIFT sont les suivantes:

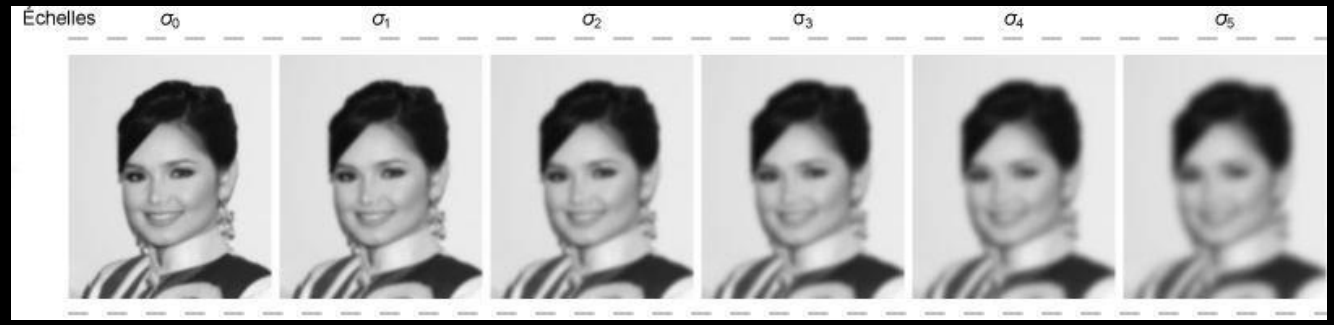
1. Détection des **extrémums** (maximums et minimums) de **différence des gaussiennes** dans **l'espace des échelles** (éléments traités dans ce qui suit)
2. **Localisation** précise de ces extrémums dans l'image et dans l'espace des échelles
   a. amélioration de la précision de localisation par **interpolation**
   b. Élimination des pixels de **faible contraste**
   c. Élimination des pixels situés sur les **arêtes**
3. Assignation d'une **orientation** préférentielle aux extrémums
4. Construction du **descripteur** de pixels d'intérêt ("feature points")

ETAPE 1 : Construction de la pyramide d'images filtrées par des gaussiennes.

Construire l'espace d'échelles $L(x,y,\sigma)$ de l'image en

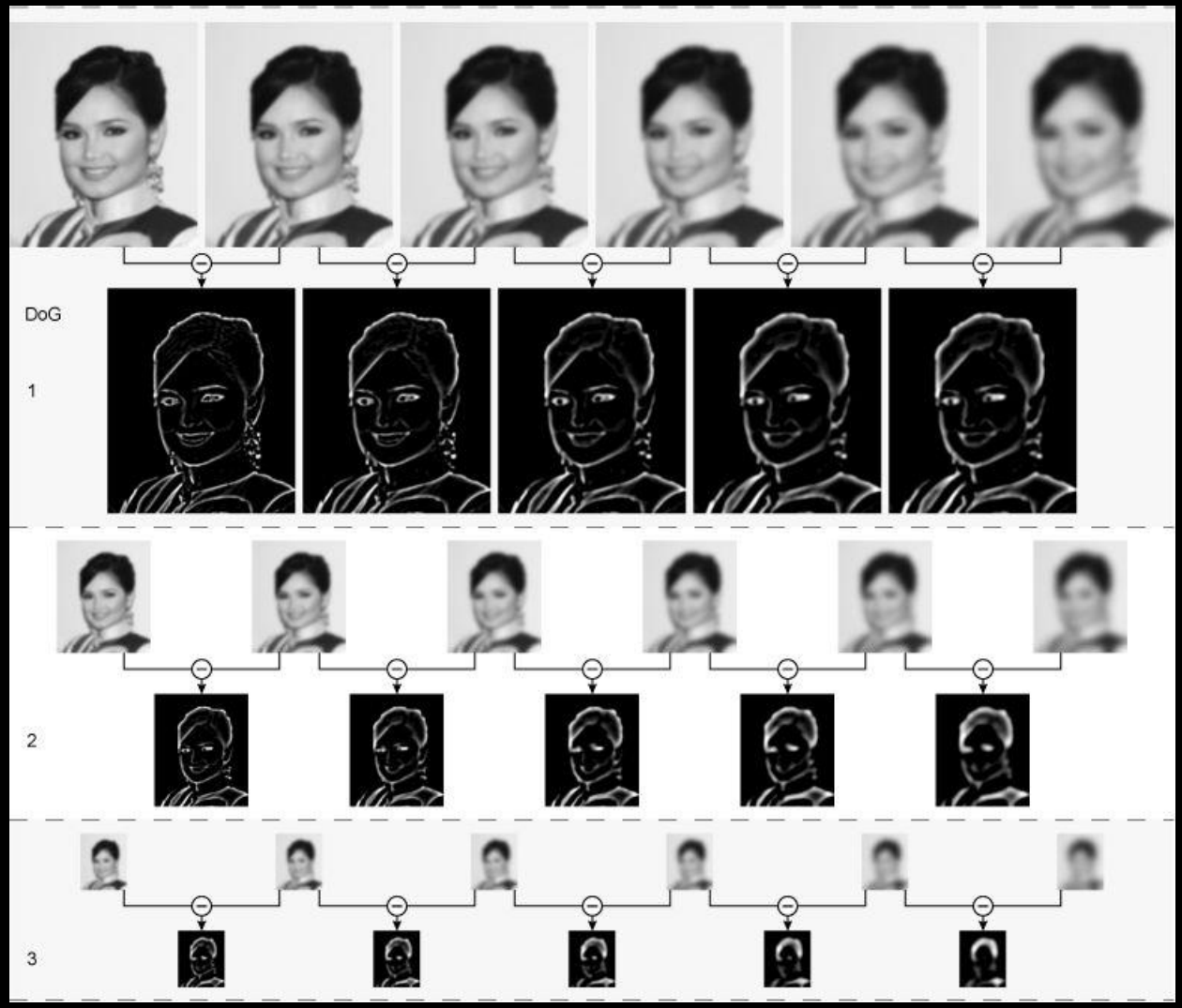
- Filtrant l'image d'origine avec un filtre gaussien d'écart-type (échelle) σ
- Générant chaque nouvelle échelle comme un facteur constant de l'échelle précédente (i.e. $\sigma_2 = k\,\sigma_1$)
- Plus l'échelle **augmente**, plus les **détails** fins (plus petits que σ) **s'estompent**

$$L(x, y, \sigma) = G(x, y, \sigma) * E(x, y) \qquad (20)$$



On construit une **pyramide** en soustrayant les images de deux échelles voisines (différences de gaussiennes DoG→ D(x,y,σ))

On effectue cette opération sur plusieurs **octaves** (i.e. un octave est situé à 2σ de l'échelle précédente)



Les paramètres utilisés pour construire la pyramide sont les suivants:

1. octave : multiplication par un facteur 2 de l'écart type d'une échelle. On a donc les octaves σ, 2σ, 4σ, 8σ
2. s : nombre d'intervalles dans un octave
3. n_e : nombre d'échelles dans un octave
4. k : facteur multiplicatif de l'écart-type d'une échelle pour obtenir l'écart-type de l'échelle suivante

D'après les analyses expérimentales menées par Lowe, les paramètres optimaux de l'approche pyramidale sont les suivants:

1. nombre d'octaves : 4
2. nombre d'échelles par octave $n_e = 5$
3. nombre d'intervalles dans un octave s = 2
4. nombre d'images par octave $n_i = s + 3 = 5$
5. facteur multiplicatif de l'écart-type d'une échelle $k = 2^{1/s}$, donc $k = \sqrt{2}$

Les diagrammes qui suivent illustrent comment l'espace d'échelles est construit.

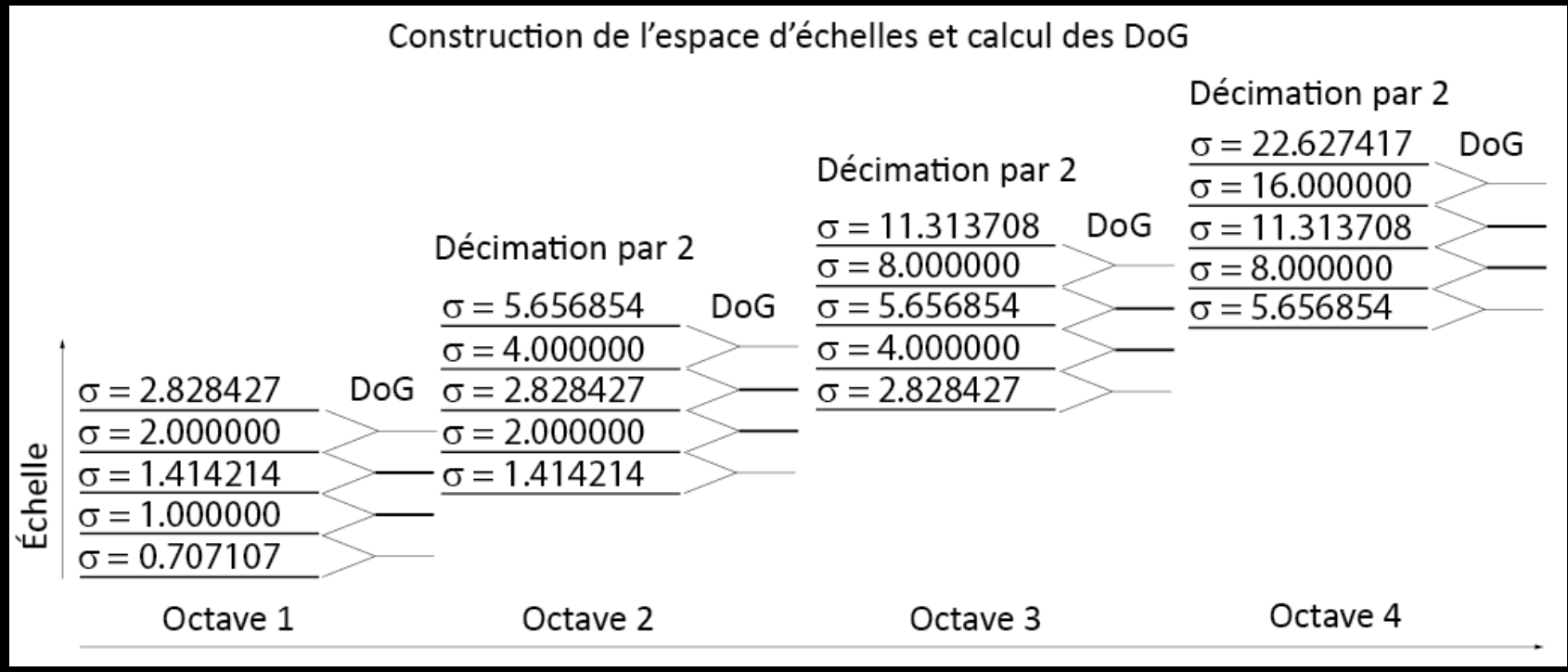
Diagramme illustrant la construction de l'espace d'échelles et le calcul des DoG

À l'octave 1, 5 images filtrées $L(x,y,\sigma)$ sont générées avec le σ de l'échelle 2 = k σ, etc.

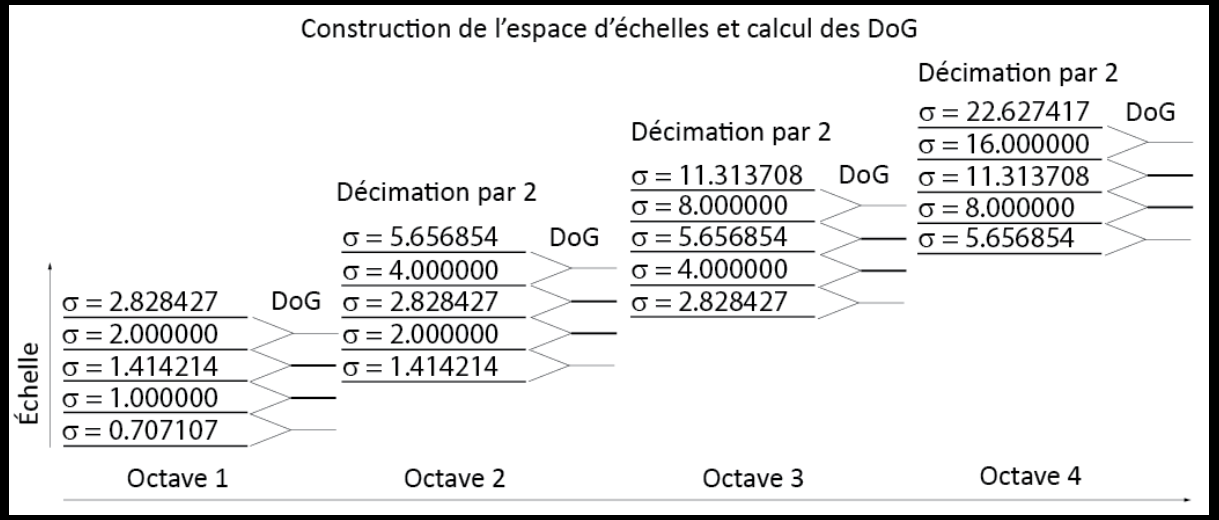
5 échelles dans l'octave permettent de calculer 4 DoG et de générer deux images DoG ayant au moins une image au-dessus et une image en dessous pour permettre de détecter des extrémums (voir la suite).

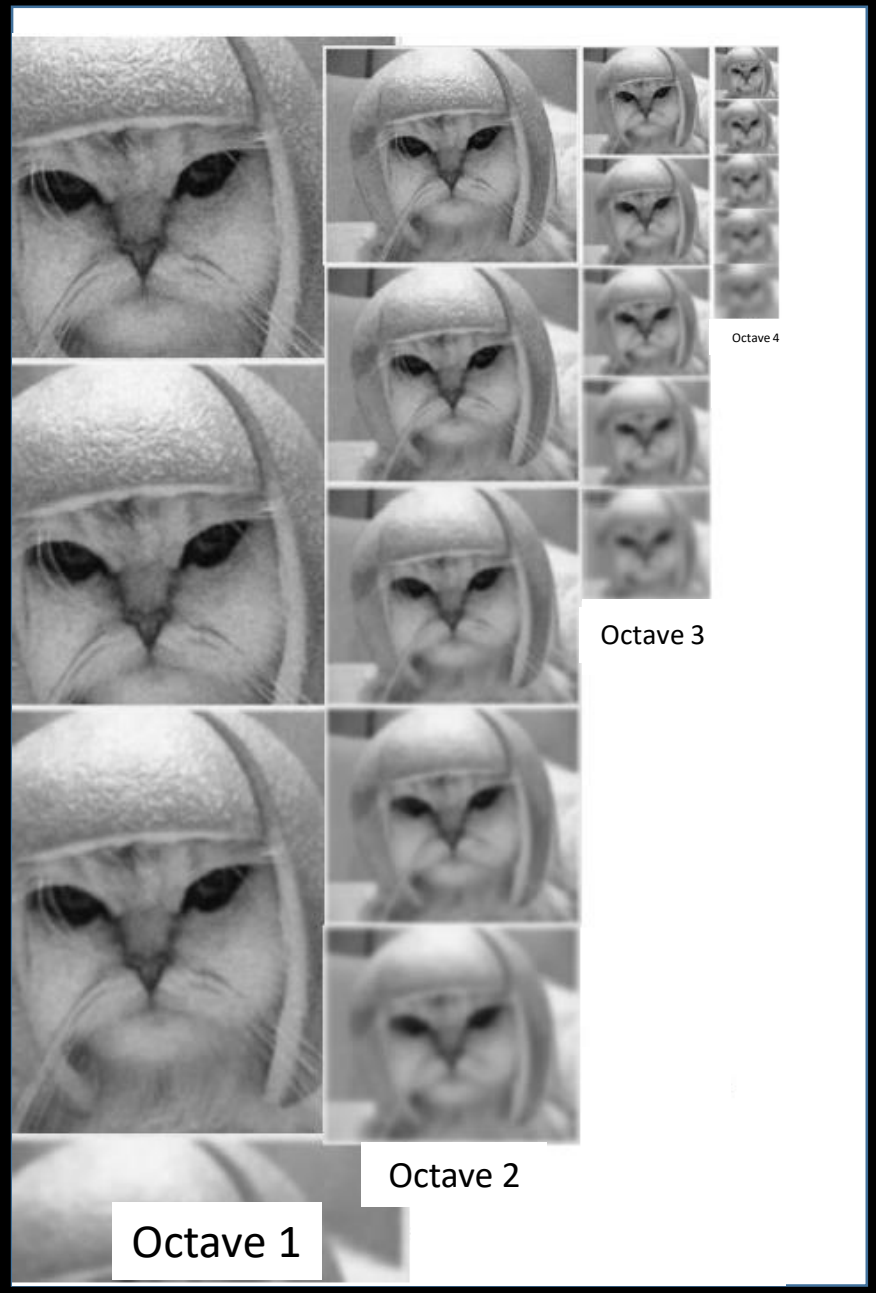
À l'octave 2, on récupère l'image "au centre" de l'octave 1, on la décime par 2 sur les lignes et les colonnes et on filtre.

On procède ainsi pour les octaves 3 et 4



Résultat de la construction de la pyramide de différences de gaussiennes

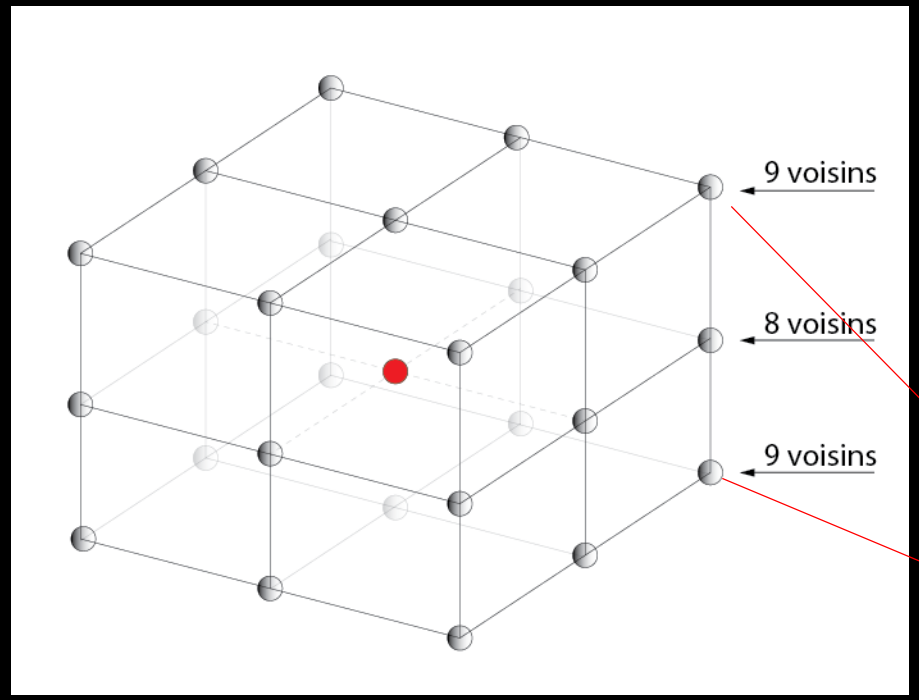




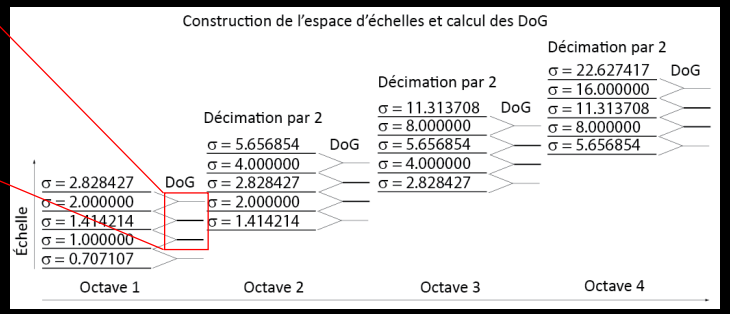
Tiré de : http://aishack.in/tutorials/sift-scale-invariant-feature-transform-scale-space/

ETAPE 2 : Détection des extrémums de différences de gaussiennes dans l'espace des échelles

Un pixel d'intérêt "candidat" de coordonnées (x,y,σ) est un pixel où la différences de gaussiennes (DoG) est un extrémum (maximum ou minimum) par rapport à ses 26 voisins immédiats



$$\left\{D(x+\partial_x, y+\partial_y, s\sigma), \partial_x \in \{-1,0,1\}, \partial_y \in \{-1,0,1\}, s \in \{k-1,0,k\}\right\} \quad (21)$$



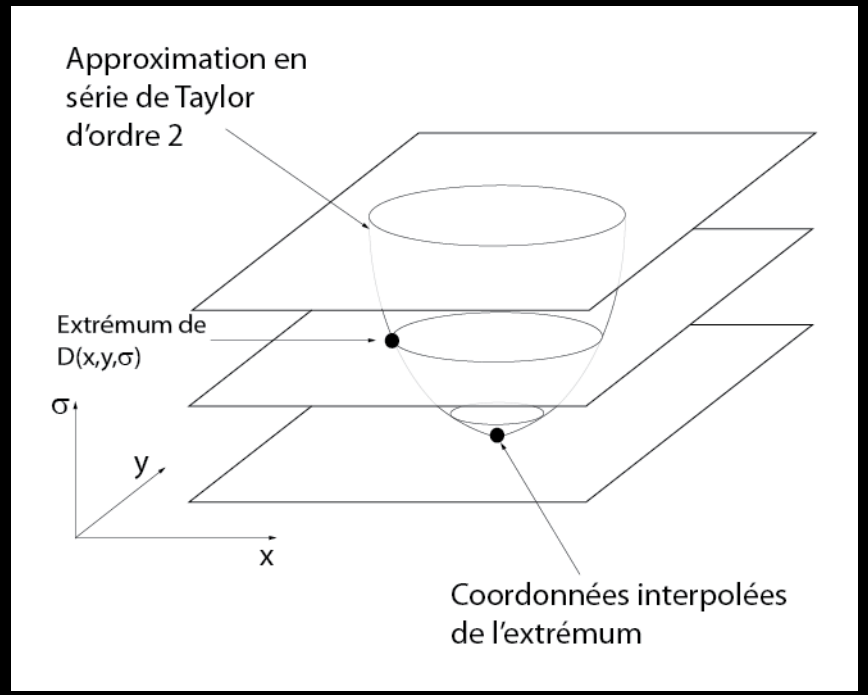
Exemple de résultat de pixels d'intérêt potentiels

On remarque que plusieurs pixels d'intérêt potentiels sont dans des régions de l'images qui sont peu significatives (le ciel par exemple) et sont par conséquent peu stables. Ils doivent être éliminés

On souhaite également que la localisation des pixels d'intérêt potentiels soit précise (sub-pixel, sub-échelle). Par conséquent, on cherche à interpoler leur position.

Approche **d'interpolation** de la position des pixels d'intérêt potentiels:

Développement en **série de Taylor d'ordre 2** de la fonction D(x,y,σ) en prenant comme **origine** les coordonnées du **pixel candidat**



$$D(x) = D + \frac{\delta D^T}{\delta x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x \qquad (22)$$

Les dérivées sont estimées numériquement

Dans (22), "x" est un *delta* par rapport aux coordonnées du pixel candidat

La position interpolée $\hat{x}$ de l'extrémum est calculée en égalant la dérivée de (22) à zéro

$$\frac{\delta D(x)}{\delta x} = 0 \qquad (23)$$

qui donne:

$$\hat{x} = -\frac{\partial^2 D}{\partial x^2}^{-1} \frac{\delta D}{\delta x} \qquad (24)$$

Preuve de (24) en 1 dimension

$$f(x) = f + f'\,x + \frac{1}{2} f''\,x^2 \tag{25}$$

$$\frac{\delta f}{\delta x} = 0 + f''\,x + f' + \frac{1}{2} f'''\,x^2 + \frac{2}{2} f''\,x \tag{26}$$

En simplifiant et en négligeant f’’’ on obtient:

$$2f''\,x + f' = 0 \tag{27}$$

d’où la solution pour x :

$$x = -\frac{f'}{f''} \tag{28}$$

(on laisse tomber le facteur ½ car il ne sert à rien)

En revenant à (24), on a que si $\hat{x} > 0.5$ dans l'une des trois dimensions, cela signifie que le pixel est plus près d'un des voisins dans l'espace discret. Dans ce cas, le pixel d'intérêt candidat est mis à jour et l'interpolation est réalisée à partir de ces nouvelles coordonnées. Autrement, le delta est ajouté aux coordonnées du pixel candidat et sa localisation gagne en précision.

ETAPE 3 : Élimination des pixels d’intérêt de faible contraste

Un seuillage absolu appliqué sur les pixels d’intérêt candidats pour lesquels

$$|D(\hat{x})| < 0.03 \tag{29}$$

permet d’éliminer ceux qui sont instable et situés dans les zones de faible contraste

Avant le seuillage

Après le seuillage

ETAPE 4 : Élimination des pixels d'intérêt situés sur les arêtes d'illuminance

Les pixels d'intérêt situés près des arêtes d'illuminance sont **instables** car leur **apparence** dépend beaucoup du **point de vue**. Il sont aussi **instables** car ils peuvent **disparaître** selon selon certaines conditions d'éclairage.

Pour éliminer ces pixels d'intérêt près des arêtes, on procède de façon similaire à l'approche de détection de coins de Harris, soit par l'analyse d'une **matrice** construite dans le voisinage de ces pixels.

La matrice d'intérêt est la Hessienne de la fonction DoG **$D(x,y,\sigma)$**

Cette matrice permet d'étudier les **courbures principales** de $D(x,y,\sigma)$

La matrice Hessienne H au pixel d'intérêt (x,y,σ) est définie comme suit:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix} \quad (30)$$

Les **valeurs propres** de H sont proportionnelles aux **courbures principales** de D(x,y,σ)

Nous porterons attention au **rapport des courbures** (ou des valeurs propres) de H

$$\begin{matrix} r < seuil \;\; OK \\ r > seuil \;\; arête \end{matrix} \quad (31)$$

Soit α la **plus grande valeur propre** de H et β **la plus petite**. On peut calculer la **trace** de H notée *Tr(H)* et le **déterminant** de H noté *Det(H)* :

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta \tag{32}$$

$$Det(H) = D_{xx}D_{yy} - D_{xy}^2 = \alpha\ \beta \tag{33}$$

Le rapport r entre les valeurs propres est donné par:

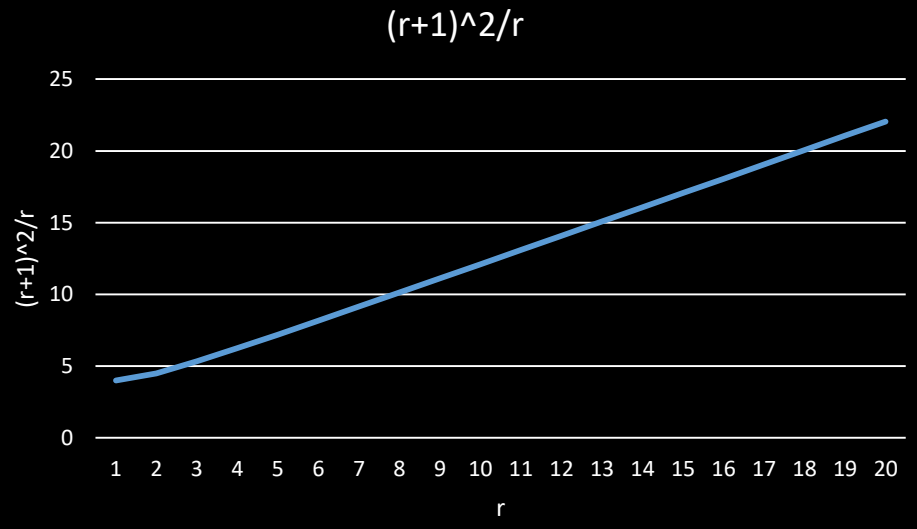
$$r = \frac{\alpha}{\beta} \tag{34}$$

En utilisant (34) et en calculant le rapport entre $Tr^2(H)$ et Det(h) on a:

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2\beta^2}{r\beta^2} = \frac{(r+1)^2}{r} \quad (35)$$

On remarque que le rapport $\frac{Tr(H)^2}{Det(H)}$ ne dépend que du rapport des valeurs propres de H

$\frac{(r+1)^2}{r}$ est **minimum** pour r = 1.



Les analyses sur un grand nombre d'images de natures diverses ont permis de choisir r = 10 comme seuil. Par conséquent, si

$$\frac{Tr(H)^2}{Det(H)} < \frac{(10+1)^2}{10} = 12.1 \tag{36}$$

on choisit de **conserver** le pixel d'intérêt.

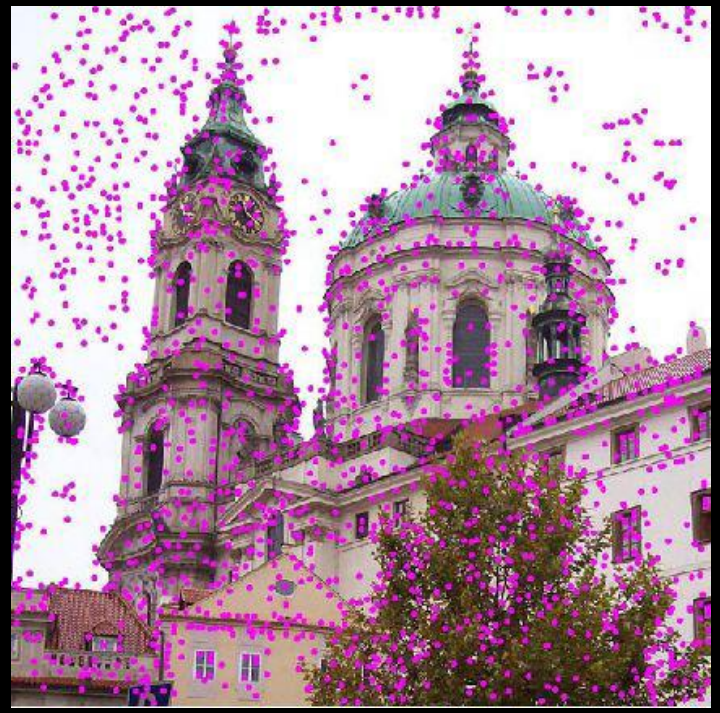
Pour

$$\frac{Tr(H)^2}{Det(H)} > 12.1 \tag{37}$$

le point d'intérêt est **éliminé**.

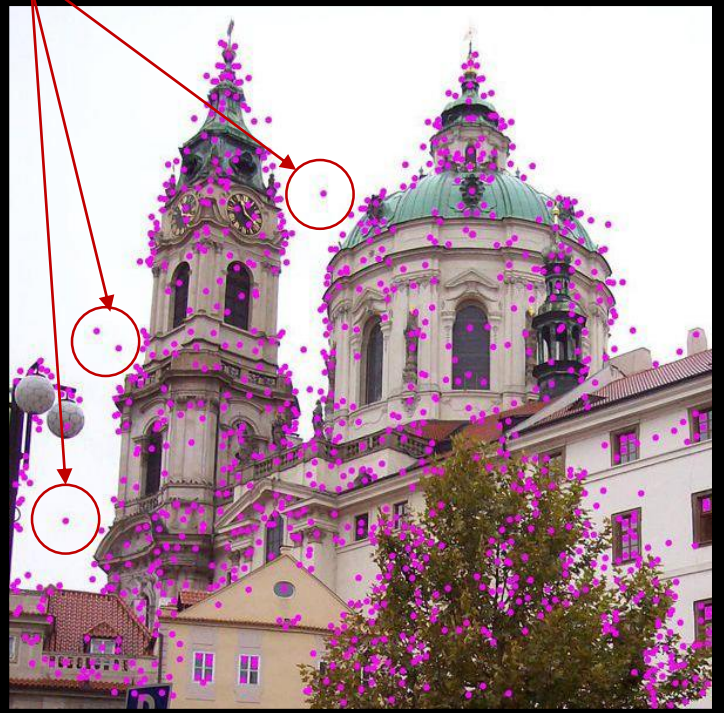
Résultat

Certains pixels ont la vie dure



Pixels d’intérêt

Pixels d’intérêt après l’élimination des pixels à faible contraste



Pixels d’intérêt après l’élimination des pixels à faible contraste et des pixels d’arête

ETAPE 5 : Assignation d'orientation privilégiée aux pixels d'intérêt

En assignant une **orientation dominante** à un pixel d'intérêt, il est par la suite possible de définir un **descripteur** pour le pixel qui soit exprimé relativement à cette orientation, ce qui rend le descripteur **robuste** aux **changements d'orientation** dans les images.

Pour un pixel d'intérêt localisé en (x,y,σ_p), on choisit l'image filtrée $L(\sigma)$ d'échelle σ la plus près de σ_p de manière à rendre les traitements subséquents invariants à l'échelle.

ETAPE 5 : Assignation d'orientation privilégiée aux pixels d'intérêt

En assignant une **orientation dominante** à un pixel d'intérêt, il est par la suite possible de définir un **descripteur** pour le pixel qui soit exprimé relativement à cette orientation, ce qui rend le descripteur **robuste** aux **changements d'orientation** dans les images.

Pour un pixel d'intérêt localisé en (x,y,σ_p), on choisit l'image filtrée $L(\sigma)$ d'échelle σ la plus près de σ_p de manière à rendre les traitements subséquents invariants à l'échelle.

Pour **chaque pixel** (x,y) de L(x,y,σ), on calcule **l'amplitude** m(x,y) du **gradient**:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + \big(L(x,y+1) - L(x,y-1)\big)^2} \tag{38}$$

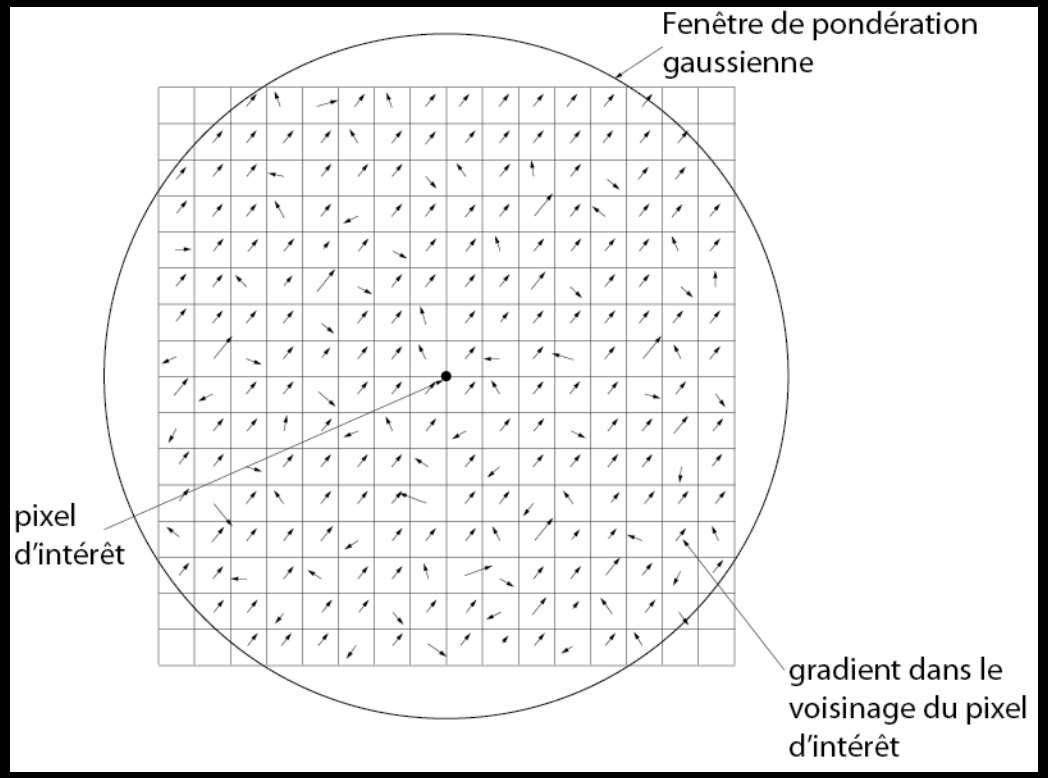
et **l'orientation** θ(x,y) du gradient:

$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{(L(x+1,y) - L(x-1,y)}\right) \tag{39}$$

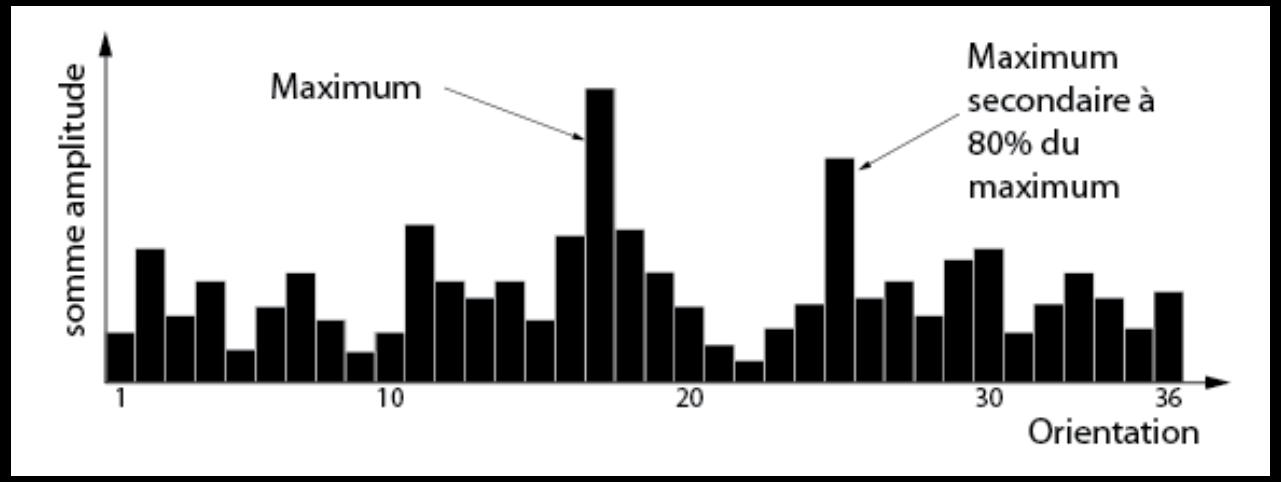
Un **histogramme d'orientations** est construit à partir des **orientations** du **gradient** des pixels **voisins** d'un **pixel d'intérêt**.

L'histogramme d'orientations comporte **36 alvéoles** ("bins") couvrant la plage complète des 360° d'orientations possibles au pixel d'intérêt

Chaque **échantillon** ajouté à l'histogramme (dans l'alvéole correspondant à l'orientation du gradient) est **pondéré** par **l'amplitude du gradient** m(x,y) et par un **poids** d'une **fenêtre circulaire** de σ égal à 1.5 fois l'échelle du **pixel d'intérêt**



L'histogramme est construit pour une région de 16 x 16 autour du pixel d'intérêt



Les **maximums** de l'histogramme correspondent à des **orientations dominantes** au point d'intérêt. Le maximum absolu de l'histogramme est détecté. L'orientation de ce maximum est assignée au pixel d'intérêt.

Les **maximums secondaires** dont l'amplitude a une valeur de **80%** de celle du maximum absolu sont aussi détectés. Pour chaque **maximum secondaire**, un **pixel d'intérêt** est aussi créé aux mêmes coordonnées (et à la même échelle) que le pixel d'intérêt correspondant au maximum.

En général, seulement **15%** des points se voient assigner plusieurs orientations. Cependant, ils contribuent significativement à la **robustesse** de l'approche.

ETAPE 6 : Construction du descripteur SIFT

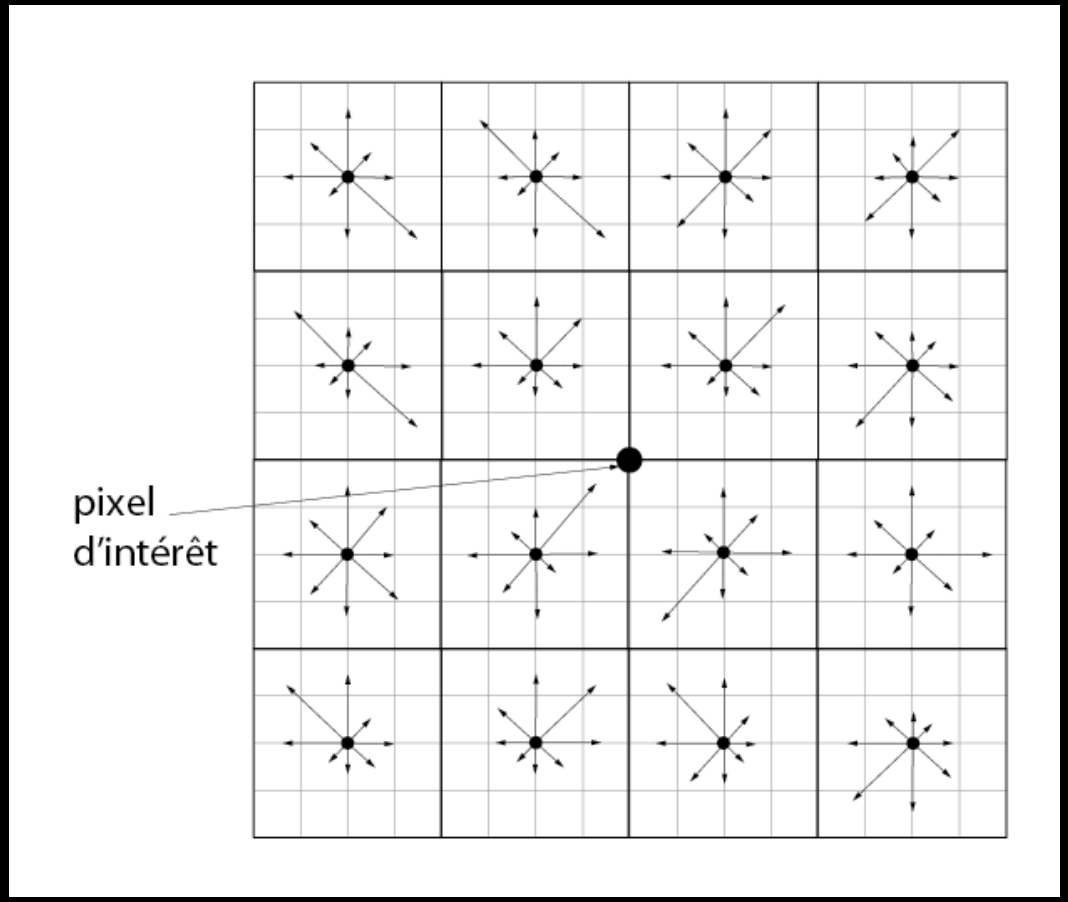
Dans les étapes précédentes, des points d'intérêt ont été repérés. Ces points se sont vu assigner:

1. une **position (x,y)**
2. une **échelle** σ
3. une **orientation dominante** entre 0 et 360°.

Ces paramètres imposent un système de coordonnées 2D par rapport auquel on peut décrire l'image localement et fournir une invariance par rapport à ceux-ci.

Ces points d'intérêt **décrivent une région de l'image** (dans l'espace des échelles) de manière **stable** et **invariante** par rapport aux paramètres de localisation, d'échelle et d'orientation.

Le descripteur du point d'intérêt est construit avec des sous-régions 4 x 4 du voisinage 16 x 16 du point d'intérêt.



Un histogramme d'orientations (8 alvéoles) est construit pour chaque sous-région 4 x 4. Dans le schéma de gauche, la longueur d'une flèche est proportionnelle à la somme de l'amplitude des gradients dans l'alvéole correspondant à sa direction (référée à l'orientation privilégiée).

Comme la région 16 x 16 est centrée sur le point d'intérêt, les éléments de la région tombent entre les pixels. On interpole les valeurs de gradients par une interpolation bilinéraire pour construire les histogrammes.

Le **descripteur** est formé en concaténant **les entrées des 8 alvéoles des 4 x 4** sous-regions pour former un **vecteur** de 4 x 4 x 8 = **128 éléments**

De plus, pour réduire les effets de changement d'éclairage, le vecteur de 128 éléments est normalisé pour le rendre unitaire (i.e. module de 1).

Finalement, pour réduire les effets de la **saturation** de certaines zones de l'image et pour réduire les **effets non-linéaires du changement d'éclairage** produisant des valeurs de gradient élevées, un **seuil de 0.2** est appliqué aux éléments du vecteur normalisé. Les alvéoles dépassant ce seuil sont mises à zéro et le vecteur résultant est renormalisé pour avoir une norme unitaire.

Le seuil de 0.2 a été déterminé expérimentalement en étudiant un grand nombre d'images.