Hiver 2018

Analyse d'images
IMN 259

Extraction de caractéristiques

Par
Pierre-Marc Jodoin

L'objectif

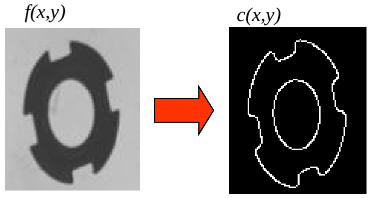
L'objectif dans cette section est d'extraire des caractéristiques ***(FEATURES)***

« Features are local, meaningful, detectable parts of an image. »
Trucco-Verri p.68

Extraction de contours

Quel est l'objectif?

f(x,y) *c(x,y)*



[Crédit Marie-Flavie Auclair-Fortier]

f(x,y) : une image d'entrée
c(x,y) : une image binaire, $c(i,j) \in \{contour, non-contour\}$

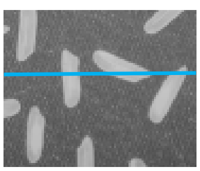
étiquettes

On appelle parfois ***edge map*** l'image c(x,y)

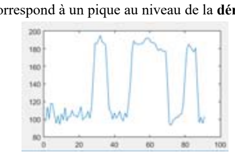
4

Une première observation

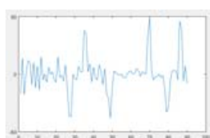
Un contour correspond généralement à une **discontinuité d'intensité**;
Une dicontinuité d'intensité correspond à un pique au niveau de la **dérivée d'ordre 1**.



Une coupe des niveaux gris dans une image à deux dimensions



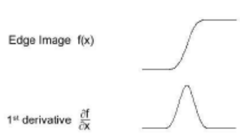
Niveaux de gris le long de la ligne bleue



Première dérivée

5

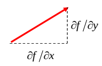
Dérivée, fonction *f(x)* (rappel)



6

Gradient, fonction *f(x,y)* (rappel)

$$\boxed{\nabla f = \begin{pmatrix} \partial f/\partial x \\ \partial f/\partial y \end{pmatrix}} \quad \begin{cases} \text{Magnitude : } \sqrt{(\partial f/\partial x)^2 + (\partial f/\partial y)^2} \\ \text{Direction : } arctan((\partial f/\partial y)/(\partial f/\partial x)) \end{cases}$$



Approximation numérique du **gradient en X**

$$\partial f/\partial x = \lim_{\Delta h \to 0} \frac{f(x+\Delta h, y) - f(x,y)}{\Delta h} = \lim_{\Delta h \to 0} \frac{f(x,y) - f(x-\Delta h, y)}{\Delta h} = \lim_{\Delta h \to 0} \frac{f(x+\Delta h, y) - f(x-\Delta h, y)}{2\Delta h}$$

Puisque le plus petit élément dans une image est le pixel de taille *1x1*, le gradient se calcul avec $\Delta h = 1$

$\partial f/\partial x \approx f(x+1,y) - f(x,y)$ "forward difference"

$\partial f/\partial x \approx f(x,y) - f(x-1,y)$ "backward difference"

$\partial f/\partial x \approx \frac{f(x+1,y) - f(x-1,y)}{2}$ "central difference"

Approximation numérique du **gradient en Y**

$\partial f/\partial y \approx f(x,y+1) - f(x,y)$ "forward difference"

$\partial f/\partial y \approx f(x,y) - f(x,y-1)$ "backward difference"

$\partial f/\partial y \approx \frac{f(x,y+1) - f(x,y-1)}{2}$ "central difference"

Gradient, fonction *f(x,y)*

Forme du filtre

$\partial f/\partial x \approx f(x+1,y) - f(x,y)$ $\quad (0 \;\; -1 \;\; 1)$

$\partial f/\partial x \approx f(x,y) - f(x-1,y)$ $\quad (-1 \;\; 1 \;\; 0)$

$\partial f/\partial x \approx \frac{f(x+1,y) - f(x-1,y)}{2}$ $\quad (-1/2 \;\; 0 \;\; 1/2)$

$\partial f/\partial y \approx f(x,y+1) - f(x,y) \rightarrow \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix}$

$\partial f/\partial y \approx f(x,y) - f(x,y-1) \rightarrow \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}$

$\partial f/\partial y \approx \frac{f(x,y+1) - f(x,y-1)}{2} \rightarrow \begin{pmatrix} -1/2 \\ 0 \\ 1/2 \end{pmatrix}$

Gradient appliqué à l'image

Exemples

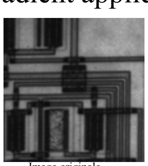
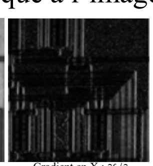
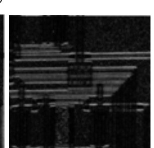


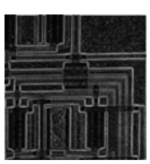
Image originale



Gradient en X : $\partial f/\partial x$



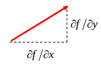
Gradient en Y : $\partial f/\partial y$



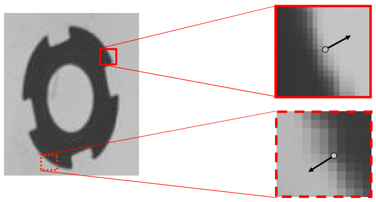
Magnitude : $\sqrt{(\partial f/\partial x)^2 + (\partial f/\partial y)^2}$

Gradient appliqué à l'image

$$\boxed{\nabla f = \begin{pmatrix} \partial f / \partial x \\ \partial f / \partial y \end{pmatrix}} \left\{ \begin{array}{l} \text{Magnitude} : \sqrt{(\partial f / \partial x)^2 + (\partial f / \partial y)^2} \\ \text{Direction} : \mathit{arctan}((\partial f / \partial y)/(\partial f / \partial x)) \end{array} \right.$$

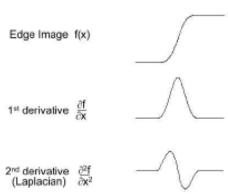


Pointe toujours en direction du changement maximal.



Note : plus la magnitude du gradient est **<u>forte</u>**, plus la discontinuité est **<u>brusque</u>**.

Laplacien (rappel)



Laplacien

Dérivée seconde et Laplacien

$$\begin{aligned} \frac{\partial^2 f(x,y)}{\partial^2 x} &= \frac{\partial}{\partial x}\left(\frac{\partial f(x,y)}{\partial x}\right) \\ &= \frac{\partial}{\partial x}\left(f(x+1/2,y) - f(x-1/2,y)\right) \\ &= \frac{\partial f(x+1/2,y)}{\partial x} - \frac{\partial f(x-1/2,y)}{\partial x} \\ &= f(x+1,y) - f(x,y) + f(x-1,y) - f(x,y) \\ &= f(x-1,y) - 2f(x,y) + f(x+1,y) \end{aligned}$$

$\boxed{\text{Masque du filtre } (1 \quad -2 \quad 1)}$

Le laplacien

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

$$\longrightarrow (1 \quad -2 \quad 1) + \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix} = \boxed{\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}}$$

Laplacien appliqué à l'image

$\frac{\partial^2 f(x,y)}{\partial^2 x}$ $\qquad$ $\frac{\partial^2 f(x,y)}{\partial^2 y}$ $\qquad$ $\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial^2 x} + \frac{\partial^2 f(x,y)}{\partial^2 y}$

Masque du filtre : $\begin{pmatrix} 1 & -2 & 1 \end{pmatrix}$ $\qquad$ $\begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}$ $\qquad$ $\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$

13

Extraction de contours

Seuil du gradient

14

Algorithme B1

Une approche bête et [parfois] méchante

Seuil d'un simple gradient

1: Calculer la magnitude du gradient en chaque point de l'image:

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2}$$

2: Calculer c(x,y) à l'aide d'un seuil

$$c(x,y) = \begin{cases} 1 & \text{si } |\nabla f(x,y)| > seuil \\ 0 & \text{sinon} \end{cases}$$

Pour minimiser les temps de calcul:

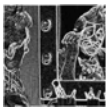
$$|\nabla f(x,y)| \approx \left|\frac{\partial f(x,y)}{\partial x}\right| + \left|\frac{\partial f(x,y)}{\partial y}\right|$$

15

Une approche bête et [parfois] méchante

Seuil d'un simple gradient

$f(x,y)$



$|\nabla f(x,y)|$

Seuil = 30

Seuil = 40

Seuil = 60

Seuil = 100

Une approche bête et [parfois] méchante

Inconvénients du seuil d'un simple gradient

1. Méthode sensible au **bruit**
2. Les contours ne sont pas **filiformes**.
3. Permet l'apparition de *edge pixels* et de *edge segments* **isolés**

Seuil = 60

Seuil = 60, avec un peu de bruit.

Conclusion: cette méthode ne fonctionne bien que pour les **images sans bruit** et avec de **fortes discontinuités**.

17

Extraction de contours

Pré-filtrage

18

Algorithme B2

Pour lutter contre le bruit : pré-filtrage

Pour réduire les effets du bruit, on peut **préfiltrer l'image** avec un filtre **passe-bas**.

Seuil d'un gradient filtré

1: Filtrer l'image d'entrée avec un filtre passe-bas:

$$g = f * h \quad \longmapsto \text{Ajout de flou}$$

2: Calculer la magnitude du gradient en chaque point de l'image:

$$|\nabla g(x,y)| = \sqrt{\left(\frac{\partial g(x,y)}{\partial x}\right)^2 + \left(\frac{\partial g(x,y)}{\partial y}\right)^2}$$

3: Calculer c(x,y) à l'aide d'un seuil

$$c(x,y) = \begin{cases} 1 & \text{si } |\nabla g(x,y)| > seuil \\ 0 & \text{sinon} \end{cases}$$

19

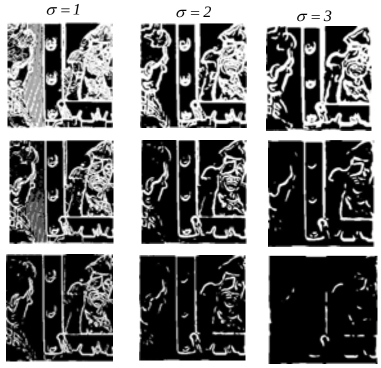
Résultats d'un préfiltrage gaussien

$\sigma = 1$ $\sigma = 2$ $\sigma = 3$

$Seuil = 10$

$Seuil = 20$

$Seuil = 30$



20

Pré-filtrage : Sobel (et Prewitt)

Plus tard dans le cours, nous introduirons deux filtres permettant d'effectuer un **filtrage passe-bas** (ajout de flou) et un **gradient** en une seule étape.

Filtre de Sobel

Filtre « gaussien » suivi d'un gradient

En Y: $$(1 \quad 2 \quad 1)/4 \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \rightarrow \boxed{\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} / 4}$$

En X: $$\begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} / 4 (-1 \quad 0 \quad 1) \rightarrow \boxed{\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} / 4}$$

Filtre de Prewitt

Filtre moyenneur suivi d'un gradient

En Y: $$(1 \quad 1 \quad 1)/3 \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \rightarrow \boxed{\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} / 3}$$

En X: $$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} / 3 (-1 \quad 0 \quad 1) \rightarrow \boxed{\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} / 3}$$

21

Algorithme B3

Détection de contours avec Sobel

Pour réduire les effets du bruit, on peut calculer le gradient de l'image à l'aide d'un filtre de Sobel (ou Prewitt).

Seuil d'un gradient calculé à l'aide du filtre de Sobel

1: Convoluer l'image d'entrée avec les deux filtres de Sobel

$$g_x = f * S_x$$
$$g_y = f * S_y$$

2: Calculer la magnitude du gradient en chaque point de l'image:

$$|\nabla g(x,y)| = \sqrt{(g_x(x,y))^2 + (g_y(x,y))^2}$$

3: Calculer c(x,y) à l'aide d'un seuil

$$c(x,y) = \begin{cases} 1 & \text{si } |\nabla g(x,y)| > seuil \\ 0 & \text{sinon} \end{cases}$$

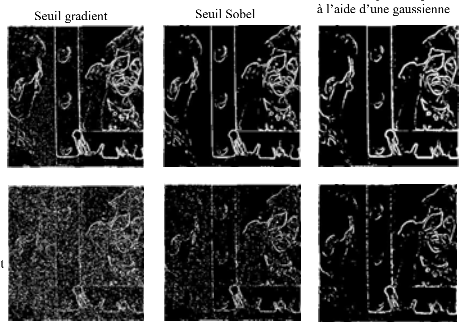
22

Pré-filtrage gaussien Vs Sobel

Seuil gradient | Seuil Sobel | Seuil d'un gradient pré-filtré à l'aide d'une gaussienne

peu de bruit

beaucoup de bruit



23

En conclusion

Seuil d'un simple gradient, sans pré-filtrage :
Bon pour les images sans bruit

Seuil d'un gradient obtenu à l'aide du filtre de Sobel :
Bon pour les images peu bruitées

Seuil du gradient d'une image pré-filtrée par un filtre gaussien :
Bon pour les images fortement bruitées

Attention! Un pré-filtrage gaussien tend à **arrondir les coins**.

24

Extraction de contours

Suppression des non-maximums

25

Extraction de contours

Inconvénients du seuil d'un simple gradient

1. Méthode sensible au **bruit**

→ **Solution :** préfiltrer l'image avec un filtre passe-bas

2. Les contours ne sont pas **filiformes**:

Que faire?

3. Permet l'apparition de *edge pixels* et de *edge segments* **isolés.** C'est ce qu'on appelle communément des *faux positifs*.

26

Suppression des non-max (*thinning*)

Rendre les contours filiformes

Bien qu'il existe une panoplie de méthodes pour rendre filiformes les contours, la plus utilisée est sans nul doute la

Suppression de non-maximums

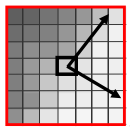
27

Suppression des non-max (*thinning*)

Rendre les contours filiformes

$f(x,y)$

Direction du contour
(perpendiculaire à la normale)



Normale au contour

$$\nabla f(x,y)=\begin{pmatrix}\partial_x f(x,y)\\ \partial_y f(x,y)\end{pmatrix}$$

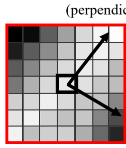
28

Suppression des non-max (*thinning*)

Rendre les contours filiformes

$|\nabla f(x,y)|$

Direction du contour
(perpendiculaire à la normale)



Normale au contour

$$\nabla f(x,y)=\begin{pmatrix}\partial_x f(x,y)\\ \partial_y f(x,y)\end{pmatrix}$$

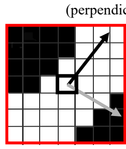
29

Suppression des non-max (*thinning*)

Rendre les contours filiformes

$|\nabla f(x,y)| > Seuil$

Direction du contour
(perpendiculaire à la normale)



Normale au contour

Contour ayant une épaisseur > 1 pixel

Pour éviter qu'un contour ait plus d'un pixel d'épais, on peut forcer à zéro tous les pixels dont $|\nabla f(x,y)|$ n'est **pas localement maximale** le long de la normale.

30

Suppression des non-max (*thinning*)

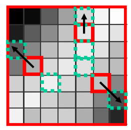
Rendre les contours filiformes

$|\nabla f(x,y)|$

Exemple de trois pixels **non-maximums**

Pixels voisins le long de la normale

Pixels non maximums



Normale

En général, on localise les deux voisins d'un pixel à l'aide **l'angle** de la normale

$$\theta(x,y) = \arctan\left(\frac{\partial f(x,y)}{\partial y} \middle/ \frac{\partial f(x,y)}{\partial x}\right)$$

31

Suppression des non-max (*thinning*)

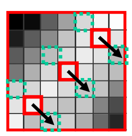
Rendre les contours filiformes

$|\nabla f(x,y)|$

Exemple de trois pixels **maximums**

Pixels voisins le long de la normale

Pixels correspondant à un maximum



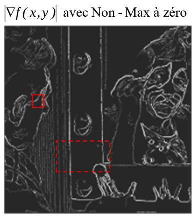
Normale

32

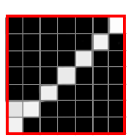
Suppression des non-max (*thinning*)

Rendre les contours filiformes

$|\nabla f(x,y)|$ avec Non - Max à zéro



Voici ce qui arrive lorsqu'on force à zéro les non-maximums



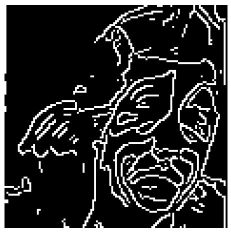
33

Suppression des non-max (*thinning*)

Rendre les contours filiformes

Voici ce qui arrive lorsqu'on seuil la norme du gradient dont les non-maximums ont été forcés à zéro.

Image binaire avec des contours d'épaisseur = 1 pixel



34

Algorithme B4

Algorithme de détection de contours par *thinning*

1: Convoluer l'image d'entrée à l'aide d'un filtre gaussien

$$g = G_\sigma * f$$

2: Calculer la magnitude du gradient en chaque point de l'image:

$$|\nabla g(x,y)| = \sqrt{\left(\frac{\partial g(x,y)}{\partial x}\right)^2 + \left(\frac{\partial g(x,y)}{\partial y}\right)^2}$$

3: Calculer l'orientation de la normale du gradient en chaque point de l'image

$$\theta(x,y) = \arctan\left(\frac{\partial g(x,y)}{\partial y} \bigg/ \frac{\partial g(x,y)}{\partial x}\right)$$

4: À l'aide de $|\nabla g(x,y)|$ et $\theta(x,y)$ supprimer les non-maximums

$$h(x,y) = \text{NonMaxSupp}(|\nabla g(x,y)|, \theta(x,y))$$

5: Appliquer un seuil

$$c(x,y) = \begin{cases} 1 & \text{si } h(x,y) > \textit{seuil} \\ 0 & \text{sinon} \end{cases}$$

35

Extraction de contours

Algorithme de Canny

36

Extraction de contours

Inconvénients du seuil d'un simple gradient

1. Méthode sensible au **bruit**

Solution : préfiltrer l'image avec un filtre passe-bas

2. Les contours ne sont pas **filiformes**:

Solution : Suppression des non-maximums

3. Permet l'apparition de *edge pixels* et de *edge segments* **isolés.**
C'est ce qu'on appelle communément des *faux positifs*.

Que faire?

37

Canny *edge detector*

C'est un des algorithmes de détection de contours parmi les plus utilisés. Cet algorithme possède trois caractéristiques:

1. Est robuste au **bruit.** Pour y arriver, il préfiltre l'image à l'aide d'un **filtre gaussien.**

2. Détecte des contours **filiformes**. Pour y arriver, il supprime les **non-maximums**.

3. Élimine les *edge pixels* isolés et les *edge segments* trop petits**.** Pour ce faire, il applique un ***seuillage*** **par hystérésis**.

38

Canny *edge detector*

Pour Canny, un contour doit respecter les conditions suivantes :

1- suite **contiguë** de pixels;

2- un contour doit avoir une épaisseur **d'un seul pixel**;

3- les pixels d'un contour doivent avoir un gradient d'intensité $|\nabla f(x,y)|$ **localement maximal**;

4- pour **TOUS** les pixels appartenant à un contour, $|\nabla f(x,y)| > Seuil_{\min}$

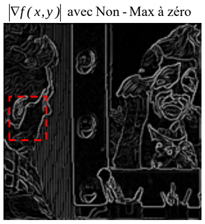
5- **AU MOINS UN** pixel appartenant au contour doit : $|\nabla f(x,y)| > Seuil_{\max}$

seuillage par hystérésis

39

Seuillage par hystérésis

$|\nabla f(x,y)|$ avec Non - Max à zéro



Seuil = 10 Seuil = 75 Seuil par hystérésis

Seuillage par hystérésis

Seuil = 10 Seuil = 75 Seuil par hystérésis

Tous les pixels ont un gradient > 10, et tous les contours ont au moins 1 pixel dont le gradient est >75.

Algorithme B5

Algorithme de détection de contours de Canny

1: Convoluer l'image d'entrée à l'aide d'un filtre gaussien

$$g = G_\sigma * f$$

2: Calculer la magnitude du gradient en chaque point de l'image:

$$|\nabla g(x,y)| = \sqrt{\left(\frac{\partial g(x,y)}{\partial x}\right)^2 + \left(\frac{\partial g(x,y)}{\partial y}\right)^2}$$

3: Calculer l'orientation de la normale du gradient en chaque point de l'image

$$\theta(x,y) = \arctan\left(\frac{\partial g(x,y)}{\partial y} \Big/ \frac{\partial g(x,y)}{\partial x}\right)$$

4: À l'aide de $|\nabla g(x,y)|$ et $\theta(x,y)$ supprimer les non-maximums

$$h(x,y) = \text{NonMaxSupp}(|\nabla g(x,y)|, \theta(x,y))$$

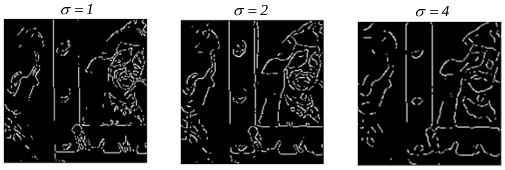
5: Appliquer un seuil par hystérésis

$$c = \text{Hysteresis}(h, |\nabla g(x,y)|, \theta(x,y), Seuil_{\min}, Seuil_{\max})$$

Canny

$Seuil_{min} = 10$, $Seuil_{max} = 40$ | $Seuil_{min} = 10$, $Seuil_{max} = 75$ | $Seuil_{min} = 10$, $Seuil_{max} = 110$

$\sigma = 1$ | $\sigma = 2$ | $\sigma = 4$

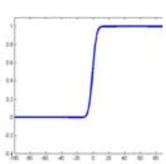


Extraction de contours

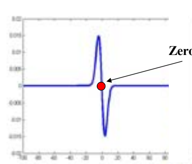
Zero-crossing du Laplacien

44

Zero-Crossing du Laplacien



Contour 1D



Dérivée seconde du contour

Il y a un contour là où la dérivée seconde croise zéro.

45

Algorithme B6

Zero-Crossing du Laplacien

Algorithme du *zero-crossing*

1: Calculer le laplacien de l'image d'entrée.

$$L(x,y) = \frac{\partial^2 f(x,y)}{\partial^2 x} + \frac{\partial^2 f(x,y)}{\partial^2 y}$$

2: POUR CHAQUE pixel (x,y) de l'image FAIRE

2a. SI il existe un voisin (i, j) du pixel (x, y) telque $L(i,j) > 0$ ET $L(x,y) \le 0$ ALORS

$c(x,y) = 1$

SINON

$c(x,y) = 0$

46

Zero-Crossing du Laplacien

Le zero-crossing est *TRÈS* sensible au bruit.

47

Zero-Crossing avec LoG *(Laplacian of Gaussian)*

Au lieu de prendre le laplacien de l'image d'entrée :

$$L(x,y) = \frac{\partial^2 f(x,y)}{\partial^2 x} + \frac{\partial^2 f(x,y)}{\partial^2 y}$$

On calcule le laplacien de l'image filtrée par une gaussienne

$$g(x,y) = G_\sigma * f$$

$$L_G(x,y) = \frac{\partial^2 g(x,y)}{\partial^2 x} + \frac{\partial^2 g(x,y)}{\partial^2 y}$$

⇒ Filtre de Marr-Hildreth (sera vu à la section sur le filtrage)

48

Zero-Crossing du Laplacien

Algorithme B7

Algorithme du zero-crossing

1. Convoluer l'image d'entrée à l'aide d'une gaussienne

$$g = G_\sigma * f$$

2. Calculer le laplacien de l'image convoluée.

$$L(x,y) = \frac{\partial^2 g(x,y)}{\partial^2 x} + \frac{\partial^2 g(x,y)}{\partial^2 y}$$

3. POUR CHAQUE pixel (x,y) de l'image FAIRE

3a. SI il existe un voisin (i, j) du pixel (x, y) telque

(*) $L(i,j) < 0$ ET $L(x,y) \geq 0$ ALORS

$c(x,y) = 1$

SINON

$c(x,y) = 0$

Note : pour rendre l'algorithme encore plus robuste au bruit, on peut remplacer la ligne (*) par

$L(i,j) < -Seuil$ ET $L(x,y) \geq Seuil$

49

Zero-Crossing avec LoG *(Laplacian of Gaussian)*

$\sigma = 1$ $\sigma = 2$ $\sigma = 3$

Il est aussi possible de mettre un **seuil supérieur à zéro** pour ne conserver que les transitions franches

50

Zero-Crossing avec LoG *(Laplacian of Gaussian)*

Avantages:
- La méthode de base exige aucun seuil
- Retourne toujours des contours filiformes
- Simple et rapide

Inconvénient:
- Ne tient pas compte de l'orientation des contours
  → effet *spaghetti*

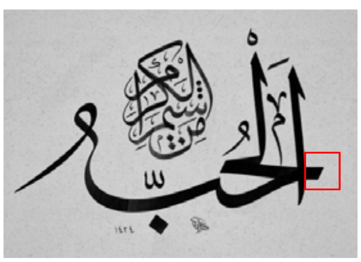
51

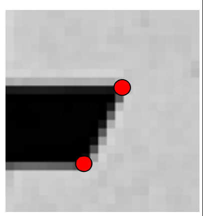
Extraction de coins

52

Détecteur de coin

Qu'est-ce qu'un coin? Une région de l'image comprenant une discontinuité ponctuelle.

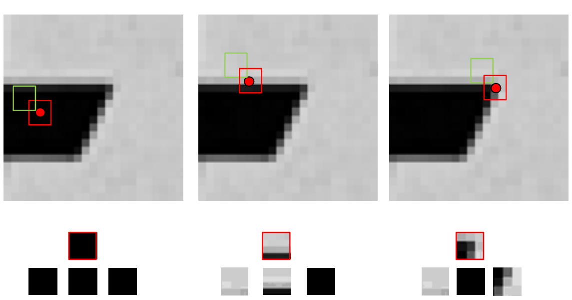




53

Détecteur de Moravec (1980)

Un coin = tout pixel dont la région est localement unique.



Plusieurs regions ont le même contenu que la fenêtre de référence

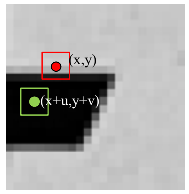
Au moins une region a le même contenu que la fenêtre de référence

Aucune region n'a le même contenu que la fenêtre de référence

Détecteur de Moravec

Soient deux regions centrées sur **(x,y)** et **(x+u,y+v)**

R_{xy} $R_{x+u,y+v}$



La difference entre les 2 s'exprime comme suit :

$$\sum_{u=-N}^{N}\sum_{v=-N}^{N}\left\|R_{xy}-R_{x+u,y+v}\right\|^2$$

Le détecteur de Moravec a pour objectif de trouver le décalage (u,v) pour lequel **la différence est la plus faible.**

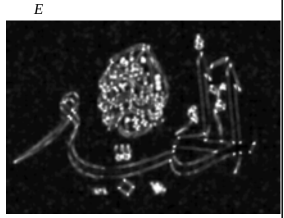
$$\boxed{E(x,y)=\min_{u,v}\sum_{u=-N}^{N}\sum_{v=-N}^{N}\left\|R_{xy}-R_{x+u,y+v}\right\|^2}$$

55

Détecteur de Moravec

f

E



56

Algorithme B8

Détecteur Moravec

Détecteur de Moravec

POUR CHAQUE pixel (x,y) de l'image *f* FAIRE

$E(x,y)$ = infini
POUR CHAQUE décalage (u,v) FAIRE

$$tmp=\sum_{u=-N}^{N}\sum_{v=-N}^{N}\left\|R_{xy}-R_{x+u,y+v}\right\|^2$$

$$E(x,y)=MIN(tmp,E(x,y))$$

$C = E > Seuil$

C : image binaire fait de points isolés.

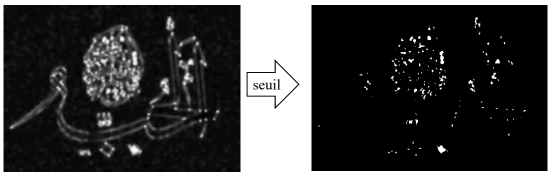
57

Détecteur de Moravec

Problème : on aimerait avoir un pixel par coin et non une region de pixels par coin.

Solution : suppression des non maximum.

E *C*



58

Algorithme B9

Suppression des non maximum (Moravec)

E = Algorithme Moravec appliqué à *f* /* B8 */

POUR CHAQUE pixel *(x,y)* de l'image *E* FAIRE

POUR CHAQUE pixel (x+u,y+v) voisin de (x,y) FAIRE

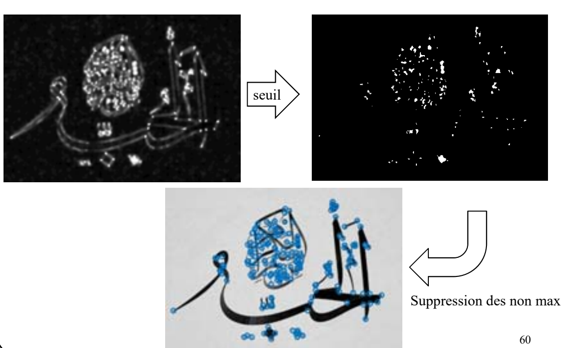
SI *E(x+u,y+v)>E(x,y)* ALORS
E(x,y) = 0

C = E > Seuil

C: image binaire fait de points isolés.

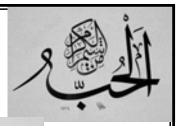
59

Détecteur de Moravec



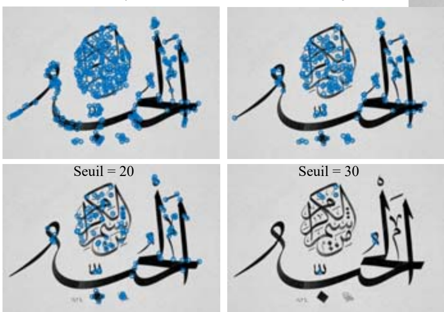
60

Détecteur de Moravec



Seuil = 10 | Seuil = 15

Seuil = 20 | Seuil = 30

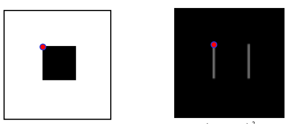


61

Détecteur de Harris (1989)

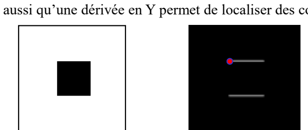
Généralisation de Moravec

On sait qu'une dérivée en X permet de localiser des contours **verticaux**



$f(x,y)$ $\left(\frac{\partial f(x,y)}{\partial x}\right)^2$

On sait aussi qu'une dérivée en Y permet de localiser des contours **horizontaux**



$f(x,y)$ $\left(\frac{\partial f(x,y)}{\partial y}\right)^2$

62

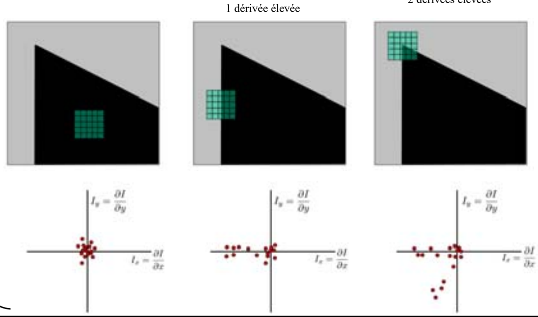
Détecteur de Harris (1989)

Généralisation de Moravec

Régions uniformes
2 dérivées faibles

Près d'un contours:
1 dérivée faible et
1 dérivée élevée

Près d'un coin:
2 dérivées élevées



Détecteur de Harris (1989)

Généralisation de Moravec

Critère de Moravec $\Rightarrow$ $\boxed{\min_{u,v} \sum_{u=-N}^{N} \sum_{v=-N}^{N} \left\| R_{xy} - R_{x+u,y+v} \right\|^2}$

Avec une extension en série de Taylor, on peut démontrer que

$$\sum_{u=-N}^{N} \sum_{v=-N}^{N} \left[f(x,y) - f(x+u, y+v) \right]^2 \approx \sum_{u=i-N}^{N} \sum_{v=j-N}^{N} \left[\frac{\partial f}{\partial x} u + \frac{\partial f}{\partial y} v \right]^2$$

$$= \sum_{u=-N}^{N} \sum_{v=-N}^{N} \left(\frac{\partial f}{\partial x}^2 u^2 + \frac{\partial f}{\partial x} \frac{\partial f}{\partial y} uv + \frac{\partial f}{\partial y}^2 v^2 \right)$$

64

Détecteur de Harris (1989)

Généralisation de Moravec

$$\sum_{u=-N}^{N} \sum_{v=-N}^{N} \left[f(x,y) - f(x+u, y+v) \right]^2 = \sum_{u=-N}^{N} \sum_{v=-N}^{N} \left(\frac{\partial f}{\partial x}^2 u^2 + \frac{\partial f}{\partial x} \frac{\partial f}{\partial y} uv + \frac{\partial f}{\partial y}^2 v^2 \right)$$

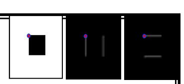
$$= \sum_{u=-N}^{N} \sum_{v=-N}^{N} (u,v) M (u,v)^T$$

$$= (u,v) \left(\sum_{u=-N}^{N} \sum_{v=-N}^{N} M \right) (u,v)^T$$

$$= (u,v) H (u,v)^T$$

$$H = \begin{pmatrix} \sum_{u=-N}^{N} \sum_{v=-N}^{N} \left(\frac{\partial f}{\partial x} \right)^2 & \sum_{u=-N}^{N} \sum_{v=-N}^{N} \frac{\partial f}{\partial x} \frac{\partial f}{\partial y} \\ \sum_{u=-N}^{N} \sum_{v=-N}^{N} \frac{\partial f}{\partial x} \frac{\partial f}{\partial y} & \sum_{u=-N}^{N} \sum_{v=-N}^{N} \left(\frac{\partial f}{\partial y} \right)^2 \end{pmatrix}$$

65



Détecteur de Harris

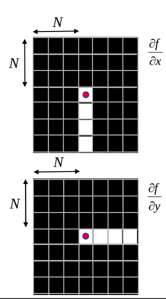
Pour Harris, un coin est une zone de l'image qui réagit fortement à la dérivée en X **ET** à la dérivée en Y.

Pour évaluer dans quelle mesure un pixel situé à la position (x,y) réagit aux deux dérivées, Harris calcule la **matrice Hessienne** (tenseur) suivante:

$$H = \begin{pmatrix} A & B \\ B & C \end{pmatrix} \text{ où } A = \sum_{x=i-N}^{i+N} \sum_{y=j-N}^{j+N} \left(\frac{\partial f(i,j)}{\partial x} \right)^2$$

$$B = \sum_{x=i-N}^{i+N} \sum_{y=j-N}^{j+N} \frac{\partial f(i,j)}{\partial x} \frac{\partial f(i,j)}{\partial y}$$

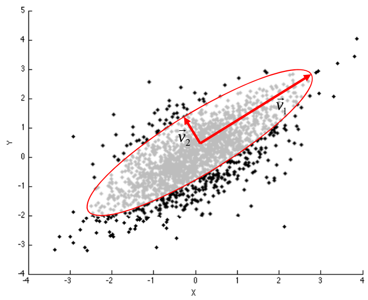
$$C = \sum_{x=i-N}^{i+N} \sum_{y=j-N}^{j+N} \left(\frac{\partial f(i,j)}{\partial y} \right)^2$$



66

Propriétés des matrices de covariance

- **vecteurs propres** $\vec{v}_1, \vec{v}_2$: principaux axes de variation
- **valeurs propres** λ_1, λ_2 :amplitude de la variation le long de ces axes

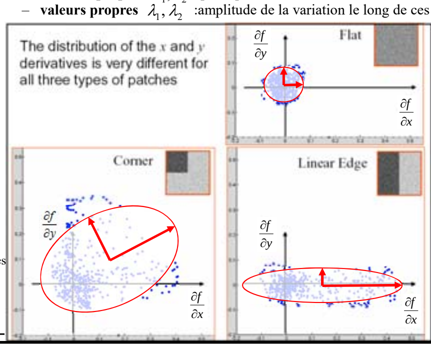


Détecteur de Harris

Crédit : Yung-Yu Chuang

Propriétés de H :

- **vecteurs propres** $\vec{v}_1, \vec{v}_2$: principaux axes de variation
- **valeurs propres** λ_1, λ_2 :amplitude de la variation le long de ces axes



$\lambda_1$et $\lambda_2$faibles

$\lambda_1$et λ_2 élevés

$\lambda_1$élevé
$\lambda_2$faible

Détecteur de Harris

Rappel: comment extraire les **valeurs propres** λ_1 et λ_2 de la matrice *H:*

$$\det(H - \lambda I) = 0$$

$$\det\left(\begin{pmatrix} A & B \\ B & C \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right) = 0$$

$$\det\begin{pmatrix} A-\lambda & B \\ B & C-\lambda \end{pmatrix} = 0$$

$$(A-\lambda)(C-\lambda) - B^2 = 0 \qquad \text{(Polynôme caractéristique)}$$

λ_1 et λ_2 Sont les racines du polynôme caractéristique

$$\lambda = \frac{A + C \pm \sqrt{(A+C)^2 + 4B^2}}{2}$$

Détecteur de Harris

$$\lambda_1 = \frac{A+C+\sqrt{(A+C)^2+4B^2}}{2}, \lambda_2 = \frac{A+C-\sqrt{(A+C)^2+4B^2}}{2}$$

A,B et C sont faibles alors $\lambda_1, \lambda_2 \approx 0$

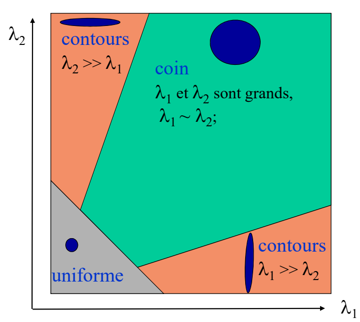
A est élevé, B et C sont faibles alors

$$\lambda_1 \approx \frac{A+\sqrt{A^2}}{2} = A$$

$$\lambda_2 \approx \frac{A-\sqrt{A^2}}{2} = 0$$

A,B et C sont élevés alors $\lambda_1, \lambda_2 >> 0$

Détecteur de Harris



Détecteur de Harris

Une fois λ_1 et λ_2 calculées on peut en conclure que

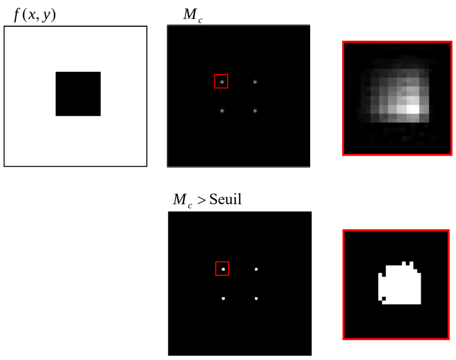
Si $\lambda_1 \approx 0$ et $\lambda_2 \approx 0$ alors le pixel est dans une <u>région uniforme</u>
Si $\lambda_1 >> 0$ et $\lambda_2 \approx 0$ alors le pixel est proche d'un <u>contour</u> ou la région est <u>fortement texturée</u>
Si $\lambda_1 >> 0$ et $\lambda_2 >> 0$ alors le pixel est proche d'un <u>coin</u>

Pour simplifier les calculs (et ainsi éviter de calculer les racines du polynôme caractéristique) Harris suggère de calculer le terme « M_c » à chaque pixel

$$\begin{aligned} M_c &= \lambda_1\lambda_2 - \kappa(\lambda_1+\lambda_2)^2 \\ &= \det(H) - \kappa \operatorname{trace}^2(H) \end{aligned}$$

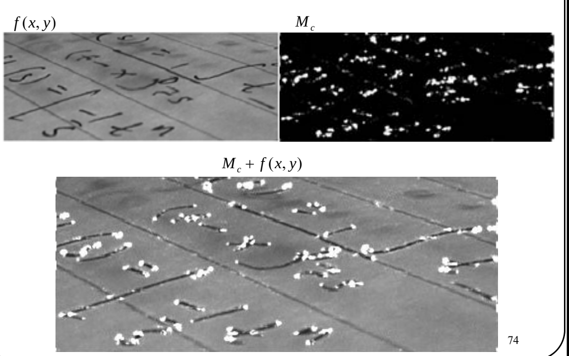
k prend généralement une valeur entre 0.02 et 0.1.

Détecteur de Harris



73

Détecteur de Harris



74

Algorithme B10

Détecteur de Harris

Détecteur de Harris avec suppression des non max

1. Calculer le gradient de l'image d'entrée

$$f_x = \frac{\partial f}{\partial x}, f_y = \frac{\partial f}{\partial y}$$

2. POUR CHAQUE pixel (x,y) de l'image FAIRE
   Calculer la matrice hessienne H autour d'un voisinage de taille $N \times N$
   $M_c(x,y) = \det(H) - \kappa \text{trace}^2(H)$

2. POUR CHAQUE pixel (x,y) de l'image FAIRE
   POUR CHAQUE pixel (x+a,y+b) voisin de (x,y) FAIRE
   SI $M_c(x+a,y+b) > M_c(x,y)$ ALORS
   $M_c(x,y) = 0$

4. $C = M_c > Seuil$

75

Détecteur de Harris

Certains auteurs prétendent qu'on peut améliorer les résultats en utilisant un **filtre gaussien**.

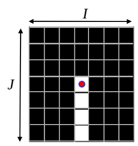
$$H = \begin{pmatrix} A & B \\ B & C \end{pmatrix}$$

$$\text{où } A = \sum_{i=x-I/2}^{I/2} \sum_{j=y-J/2}^{J/2} W_{i,j} \left(\frac{\partial f(i,j)}{\partial x} \right)^2$$

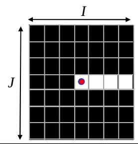
$$B = \sum_{i=x-I/2}^{I/2} \sum_{j=y-J/2}^{J/2} W_{i,j} \frac{\partial f(i,j)}{\partial x} \frac{\partial f(i,j)}{\partial y}$$

$$C = \sum_{i=x-I/2}^{I/2} \sum_{j=y-J/2}^{J/2} W_{i,j} \left(\frac{\partial f(i,j)}{\partial y} \right)^2$$

$$\boxed{W_{i,j} = \frac{1}{2\pi\sigma^2} e^{-\frac{(i-x)^2+(j-y)^2}{2\sigma^2}}}$$



$\frac{\partial f(x,y)}{\partial x}$



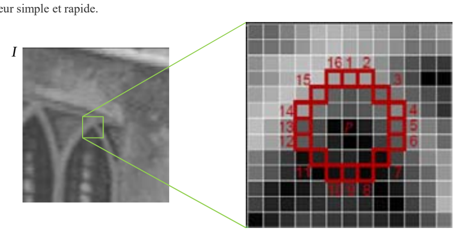
$\frac{\partial f(x,y)}{\partial y}$

76

Détecteur FAST

Features from **A**ccelerated **S**egment **T**est

Détecteur simple et rapide.



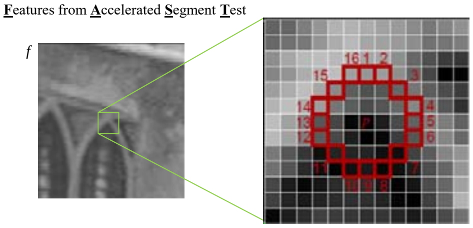
E. Rosten and T. Drummond "Fusing points and lines for high performance tracking." IEEE ICCV, 2005

credit: Ed Rosten

77

Détecteur FAST

Features from **A**ccelerated **S**egment **T**est



P=(x,y) est un coin s'il est entouré d'un arc de cercle (rayon = 4 avec 16 voisins au total) contenant plus de *N* pixels **contigus**

- Tous plus clairs que *f(x,y)*+seuil

ou

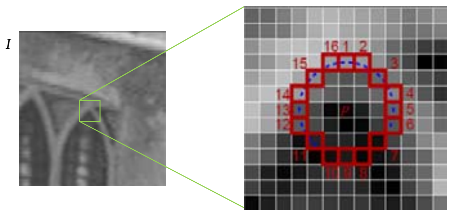
- Tous plus foncés que *f(x,y)*-seuil

78

credit: Ed Rosten

Détecteur FAST

Features from **A**ccelerated **S**egment **T**est



Ici, 12 pixels ont un niveau de gris supérieur à I(x,y)+seuil.

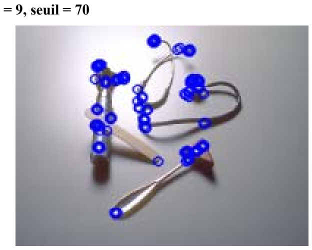
79

credit: Ed Rosten

Détecteur FAST

Features from **A**ccelerated **S**egment **T**est

Résultat pour N = 9, seuil = 70



Comme pour Harris, plusieurs points sont agglutinés les uns sur les autres.

Solution : **suppression des non maximums**.

80

Algorithme B11

Détecteur FAST

Suppression des non max

1. listCoins, nb = FAST(*f*) // FAST retourne une liste de “nb” coins

2. POUR c allant de 0 à nb-1 FAIRE
   C = listCoins[c]
   L = stocker la couleur des16 voisins de C

$$V_{inf}[c] = \sum_{\substack{i=0 \\ I(L(i))<I(C)-Seuil}}^{15} \left| I(L(i)) - I(C) \right|$$

$$V_{sup}[c] = \sum_{\substack{i=0 \\ I(L(i))>I(C)+Seuil}}^{15} \left| I(L(i)) - I(C) \right|$$

3. Supprimer tous les coins ayant un voisin dont V_{inf} ou V_{sup} est supérieur

81

Détecteur FAST

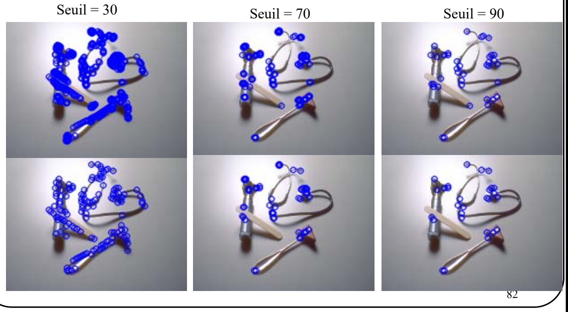
Features from **A**ccelerated **S**egment **T**est

Pour N = 9

Seuil = 30 | Seuil = 70 | Seuil = 90

NMS=False

NMS=True



82

Détecteur FAST

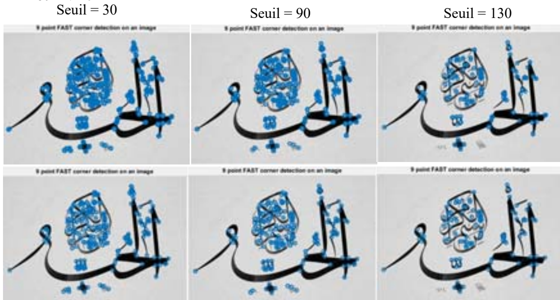
Features from **A**ccelerated **S**egment **T**est

Pour N = 9

Seuil = 30 | Seuil = 90 | Seuil = 130

NMS=False

NMS=True



83

Détecteur FAST

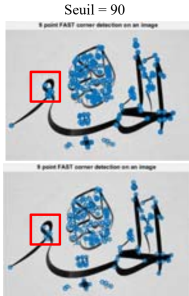
Features from **A**ccelerated **S**egment **T**est

Pour N = 9

Seuil = 90

NMS=False

NMS=True

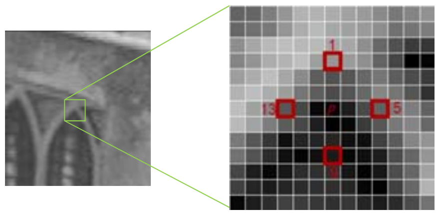


84

Détecteur FAST – version accélérée

Features from **A**ccelerated **S**egment **T**est

Tester les 4 pixels (1,5,9,13). Avec N=9, si 3 d'entre eux ne sont pas plus clairs que *I(p)*+seuil ou plus foncés que *I(p)*-seuil, **alors p n'est pas un coin**



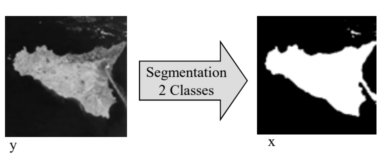
85

Segmentation

86

Définition du problème

Concept de classe et d'étiquettes



Partant d'une image d'entrée « y », on cherche à estimer une image de sortie « x » dont chaque pixel contient une **étiquette de classe** (étiquette *terre* ou étiquette *mer*).

87

Définition du problème

Tous les pixels d'une même classe partagent une (ou plusieurs) **caractéristique en commun** :

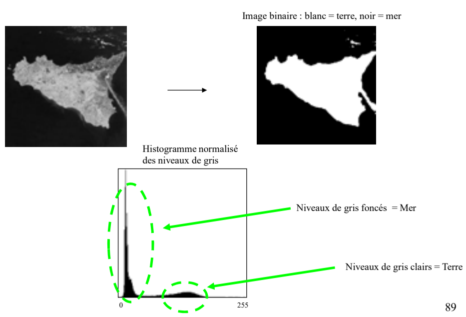
Couleur;
Niveau de gris;
Mouvement;
Texture;
(...)

Dans l'exemple précédent, on cherche à regrouper ensemble les pixels ayant un niveau de gris similaire.

88

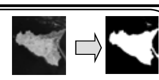
Définition du problème

Segmentation, exemple: Terre Vs Mer



89

Quelques définitions



y : un champ d'observations (image d'entrée)
x : un champ d'étiquettes (image à estimer)

Site : synonyme de pixel. Un site « s » possède les coordonnées *(i,j)* dans l'image.

$y_s = y(i,j)$ et $y_s \in [0,255]$
$x_s = x(i,j)$ et $x_s \in \{terre, mer\}$

90

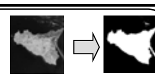
Segmentation

Algorithme du seuil

91

Algorithme B12

L'algorithme du seuil



y x

Un algorithme simple : le seuil

1. POUR CHAQUE site *s* du champ d'observations y FAIRE

SI y_s > Seuil ALORS

$x_s = 1$ /* Étiquette « *terre* » au pixel (i,j) */

SINON

$x_s = 0$ /* Étiquette « *mer* » au pixel (i,j) */

92

L'algorithme du seuil

Avantages:

- Trivial à implémenter;
- Très rapide

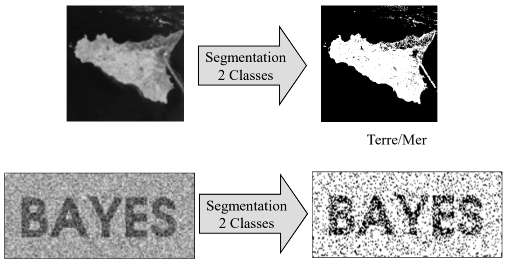
Inconvénients:

- Le seuil doit être fixé par un utilisateur
  C'est un algorithme **<u>supervisé</u>**
- Algorithme inefficace pour des images bruitées

93

L'algorithme du seuil

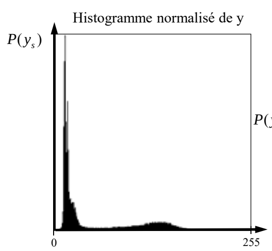
Algorithme inefficace pour des images bruitées



94

Trouver le *meilleur* seuil sans supervision

Quelques définitions



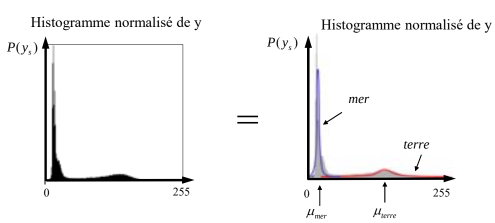
$P(y_s)$ Distribution des niveaux de gris dans l'image *y*.

$P(y_s = a)$ Peut se lire : probabilité d'observer un pixel de niveau de gris « a » dans l'image y

exemple: si $P(y_s = 15) = 0.09$ alors

si je tire au hasard un pixel dans l'image y, j'aurai 9 pourcents de chance qu'il soit d'intensité 15

Trouver le *meilleur* seuil sans supervision



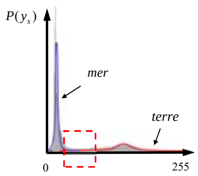
μ_{mer} : Niveau de gris moyen des pixels de la classe "mer"

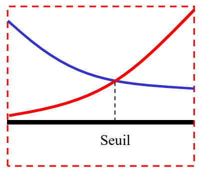
μ_{terre} : Niveau de gris moyen des pixels de la classe "terre"

96

Trouver le *meilleur* seuil sans supervision

On peut démontrer que le meilleur seuil est celui situé à l'endroit où les deux courbes se croisent.





97

Segmentation

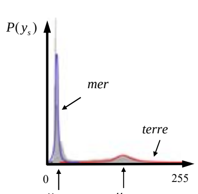
Algorithme des k-moyennes

98

Algorithme des K-moyennes (*K-means*)

L'algorithme des K-moyennes possède un double objectif:

1. Calculer le champ d'étiquettes x
2. Calculer l'intensité moyenne de chaque classe μ_{terre} et μ_{mer}



99

Algorithme des K-moyennes (*K-means*)

1. Calculer le champ d'étiquettes x

Si μ_{terre} et μ_{mer} sont connus, alors on peut facilement calculer x:

```
POUR CHAQUE site s du champ d'observations y FAIRE
    SI (y_s − μ_terre)² < (y_s − μ_mer)² ALORS
        x_s = 1      /* Étiquette « terre » au pixel (i,j) */
    SINON
        x_s = 0      /* Étiquette « mer » au pixel (i,j) */
```

100

Algorithme des K-moyennes (*K-means*)

2. Calculer μ_{terre} et μ_{mer}

Si x est connu, alors calculer μ_{terre} et μ_{mer} est facile

```
μ_mer = μ_terre = 0
POUR CHAQUE site s du champ d'observations y FAIRE
    SI x_s == 1  ALORS
        μ_terre = μ_terre + y_s
    SINON
        μ_mer = μ_mer + y_s
μ_mer = μ_mer / Nombre de pixels "Mer"
μ_terre = μ_terre / Nombre de pixels "Terre"
```

101

Algorithme B13

Algorithme des K-Moyennes

```
0. μ_mer = un pixel y_s pris au hasard
   μ_terre = un autre pixel y_s pris au hasard (μ_terre ≠ μ_mer)
1. POUR CHAQUE site s du champ d'observations y FAIRE
       SI (y_s − μ_terre)² < (y_s − μ_mer)² ALORS
           x_s = 1      /* Étiquette « terre » au pixel (i,j) */
       SINON
           x_s = 0      /* Étiquette « mer » au pixel (i,j) */
2. μ_mer = μ_terre = 0
3. POUR CHAQUE site s du champ d'observations y FAIRE
       SI  x_s == 1 ALORS
           μ_terre = μ_terre + y_s
       SINON
           μ_mer = μ_mer + y_s
4. μ_mer = μ_mer / Nombre de pixels "Mer"
   μ_terre = μ_terre / Nombre de pixels "Terre"
5. SI μ_terre et μ_mer ne changent plus arrêter SINON retour à 1
```

Estimer x (étape 1)

Recalculer μ_{terre} et μ_{mer} (étapes 2–5)

102

Algorithme des K-moyennes (*K-means*)

Avec *K-means*, on cherche à minimiser l'erreur quadratique globale qu'on appelle également la « distorsion »

$$J_{km} = \sum_{s} (y_s - \mu_{x_s})^2$$

Note : **K-means ne converge pas nécessairement** vers la solution optimale. En Effet, dépendant des paramètres de départ (μ_{mer}, μ_{terre}) l'algorithme peut converger vers des résultats différents, voire même aberrants. Pour résoudre ce problème, on peut lancer k-means plusieurs fois avec différents paramètres de départ et ne **garder que la solution qui minimise la distorsion**.

103

Algorithme des K-moyennes (*K-means*)

Algorithme des K-Moyennes pour un **nombre arbitraire de classes**

0. Initialiser la moyenne de chaque classe μ_c

1. POUR CHAQUE site s du champ d'observations y FAIRE
   x_s = étiquette de la classe dont la moyenne μ_c
   est la plus proche de y_s.

3. Recalculer la moyenne de chaque classe.

4. SI les moyennes ne changent plus ALORS arrêter SINON retour à 1

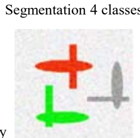
Pour en savoir plus au sujet de *K-Means* :

- Tutoriel d'andrew Moore : http://www.autonlab.org/tutorials/kmeans.html
- Livre de D.MacKay, « ***Information Theory, Inference, and Learning Algorithms*** », Chapitre 20. (http://www.inference.phy.cam.ac.uk/mackay/itprnn/book.html)

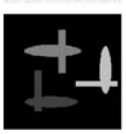
104

Algorithme des K-moyennes (*K-means*)

Segmentation 4 classes



y



x

Segmentation 5 classes

y

x

105

Algorithme des K-moyennes (*K-means*)

On peut également utiliser **d'autres caractéristiques** pour segmenter une image. On peut par exemple utiliser les caractéristiques liées à la texture.

Exemple: il est impossible de segmenter cette image en 2 classes sur la base exclusive des niveaux de gris car l'objet central possède le même histogramme que le reste de l'image.

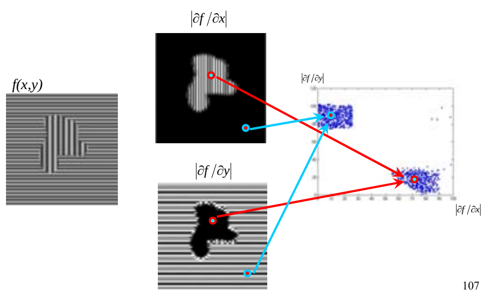
f(x,y)

Que faire? Il nous faut utiliser une autre caractéristique que le niveau de gris.

106

Algorithme des K-moyennes (*K-means*)

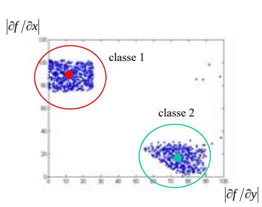
On peut par exemple utiliser les dérivées en x et en y. De cette façon, chaque pixel *(i,j)* est associé à deux valeurs : $|f_x(i,j)|$ et $|f_y(i,j)|$

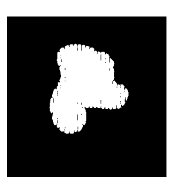


107

Algorithme des K-moyennes (*K-means*)

Maintenant que chaque pixel est associé à un point 2D, il est facile de segmenter l'image à l'aide de l'algorithme des K-moyennes.





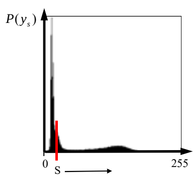
108

Segmentation

Algorithme de Otsu

109

Algorithme de Otsu



Idée: déplacer « S » de 0 à 255 afin de trouver le seuil qui maximise la **variabilité entre les deux classes:**

$$\boxed{\sigma = P(mer)P(terre)(\mu_{mer} - \mu_{terre})^2}$$

110

Algorithme de Otsu

$P(mer) = \sum_{i=0}^{S-1} P(y_s)$ (probabilité qu'un pixel appartienne à la classe "mer")

$P(terre) = \sum_{i=s}^{255} P(y_s)$ (probabilité qu'un pixel appartienne à la classe "terre")

$$\mu_{mer} = \frac{\sum_{i=0}^{S-1} \left(i \times P(y_s)\right)}{\sum_{i=0}^{S-1} P(y_s)}$$

$$\mu_{terre} = \frac{\sum_{i=s}^{255} \left(i \times P(y_s)\right)}{\sum_{i=s}^{255} P(y_s)}$$

111

Algorithme B14

Algorithme de Otsu

Algorithme de Otsu

0. $\sigma_{max} = 0;$
   $Seuil_{max} = 0;$

1. POUR S allant de 0 à 255 FAIRE
   Calculer $P(mer)$
   Calculer $P(terre)$
   Calculer μ_{mer}
   Calculer μ_{terre}
   $\sigma = P(mer)P(terre)(\mu_{mer} - \mu_{terre})^2$
   SI $\sigma > \sigma_{max}$ ALORS
   $\sigma_{max} = \sigma$
   $Seuil_{max} = S$

2. Seuiller l'image y à l'aide de $Seuil_{max}$

Note : Attention aux divisions par zéro!

112

Les faits saillants

- Détection de contours:
  Seuil d'un simple gradient
  Seuil d'un gradient obtenu avec Sobel
  Seuil d'un simple gradient avec préfiltrage gaussien

  Suppression des non-maximum

  Algorithme de Canny (seuillage par hystérésis)

  Zero crossing

- Extraction de coins
  Détecteurs de Moravec et de Harris
  Détecteur FAST
  Suppression des non maximum

- Segmentation (extraction de régions)
  Simple seuil
  Seuil probabiliste
  K-moyennes
  Otsu

113