Filtre de Canny-Deriche

ABOUZAID Mehdi
TOOMEY Damien

INSA – Institut National des Sciences Appliquées de Rouen

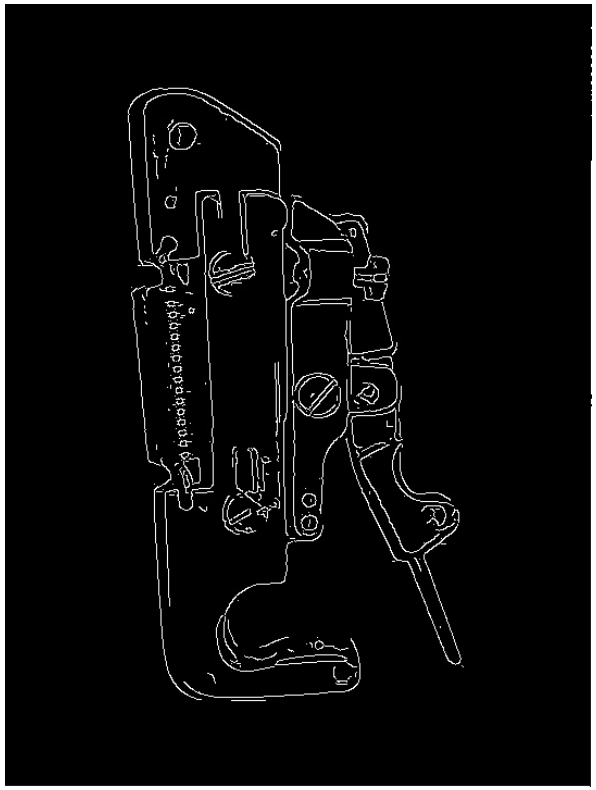
27 novembre 2018

Sommaire

Sommaire

Introduction

- Détection de contours $\rightarrow$ réduction d'information $\rightarrow$ stockage réduit
- Rachid Deriche, 1987
- Amélioration du filtre de Canny



$\alpha = 0.8$, $Smin = 26$, $Smin = 41$

Sommaire

Similitudes avec le filtre de Canny

Méthode

Détection de contours en 4 étapes, comme le filtre de Canny :

1. Gradients et lissage (réduction du bruit)
2. Norme du gradient et direction du gradient
3. Suppression des non-maxima locaux
4. Seuillage par hystérésis

Critères de détection optimale des contours

Remarque

Dans le filtre de Deriche, on retrouve également les mêmes critères de détection optimale des contours que le filtre de Canny

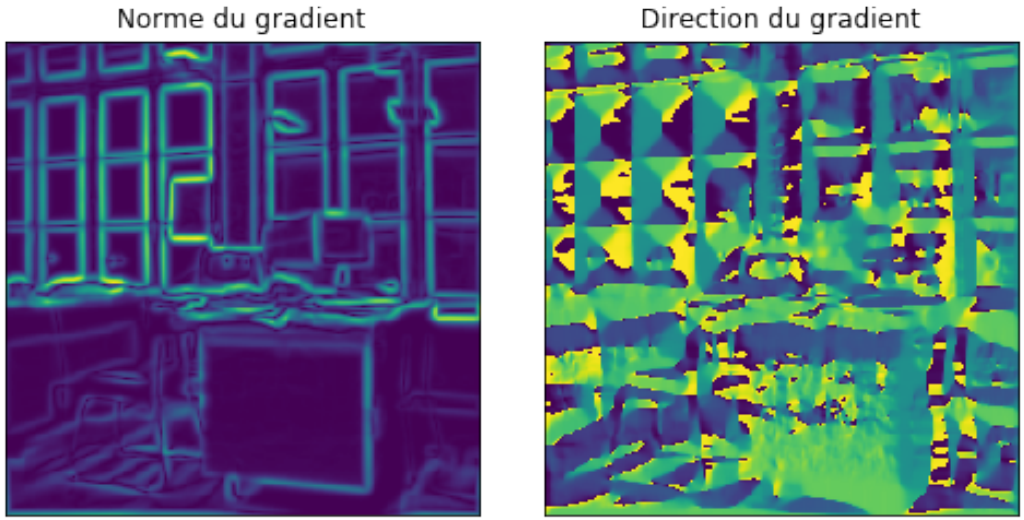
- **A** **Bonne détection** : faible taux d'erreur dans la signalisation des contours
- **B** **Bonne localisation** : les points détectés comme contours doivent être aussi proches que possible des contours réels
- **C** **Clarté de la réponse** : un point du contour ne doit être détecté qu'un seule fois par le filtre

A chaque critère (**A**,**B**,**C**) est associé une formule mathématique. La maximisation de ces trois critères conduit à la résolution de l'équation différentielle dont la solution est l'opérateur f.

Différences par rapport au filtre de Canny

La principale différence se trouve dans l'implémentation des deux premières étapes de l'algorithme :

1. Gradients et lissage (réduction du bruit)
2. Norme du gradient et direction du gradient



Contrairement au filtre de Canny, le détecteur de contours de Deriche utilise **un filtre RII (Réponse Impulsionnelle Infinie)** au lieu d'un filtre RIF (Réponse Impulsionnelle Finie)

Equation différentielle

$$2 \cdot f(x) - 2 \cdot \lambda_1 \cdot f''(x) + 2 \cdot \lambda_2 \cdot f''''(x) + \lambda_3 = 0$$

Canny : filtre RIF

Conditions initiales : $f(0) = 0, f(W) = 0, f'(0) = S$ et $f'(W) = 0$
Solution générale sur $[-W, W]$:

$$f(x) = a_1 \cdot e^{\alpha \cdot x} \cdot sin(\omega \cdot x) + a_2 \cdot e^{\alpha \cdot x} \cdot cos(\omega \cdot x) + a_3 \cdot e^{-\alpha \cdot x} \cdot sin(\omega \cdot x) + a_4 \cdot e^{-\alpha x} \cdot cos(\omega \cdot x) + C$$

Deriche : filtre RII

Conditions initiales : $f(0) = 0, f(+\infty) = 0, f'(0) = S$ et $f'(+\infty) = 0$
Solution générale sur $\mathbb{R}$:

$$f(x) = -c \cdot e^{-\alpha \cdot |x|} \cdot sin(\omega \cdot x)$$

On peut observer que la solution obtenue à l'aide du filtre RII pour le détecteur de contours de Deriche est plus simple en plus d'améliorer les résultats de détection.

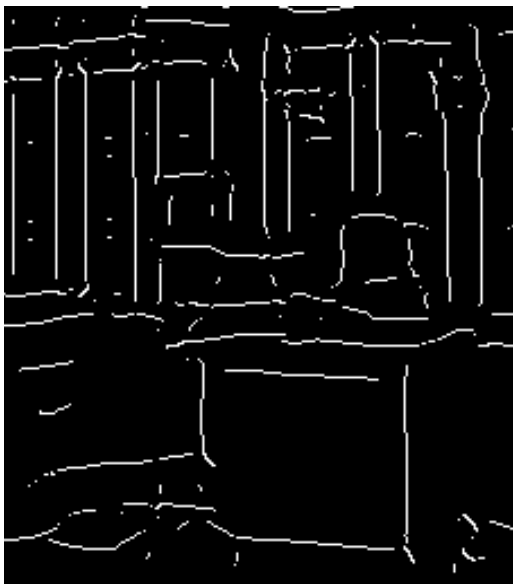
Avantage du filtre RII de Deriche

Adaptation aux caractéristiques de l'image en modifiant seulement les deux paramètres $(\alpha, \omega) \in \mathbb{R}^+$, sans impacter le temps d'exécution.

- Si la valeur de α est petite (généralement entre 0.25 et 0.5), la détection est meilleure.
- Si la valeur de α est grande (entre 2 et 3), la localisation des contours est meilleure.

C'est pourquoi, dans la plupart des cas, il est recommandé de fixer la valeur de α à 1.

α = 0.25, ω = 0.0001



α = 1, ω = 0.0001

α = 3, ω = 0.0001

Sommaire

Cas mono-dimensionnel

Deux détecteurs de contours optimaux de réponse impulsionnelle :

$$f(x) = -c \cdot e^{\alpha \cdot |x|} \cdot sin(\omega \cdot x)$$
$$g(x) = -c \cdot x \cdot e^{\alpha \cdot |x|}$$

Remarque

Les résultats de détection de contours sont améliorés en utilisant $g(x)$ au lieu de $f(x)$.
Avec $g(x)$, le filtre ne dépend que de α et non plus de ω.

Réponse impulsionnelle $h(n)$ (fonction de lissage)

$h(n)$ correspond à l'intégrale $h(x)$ de $f(x)$ avec $n \in \mathbb{Z},\ x \in \mathbb{R}$

$$h(n) = (c_1 \cdot sin(\omega \cdot |n|) + c_2 \cdot cos(\omega \cdot |n|)) \cdot e^{-\alpha \cdot |n|}$$

Coefficients du filtre

Remarque

La forme du filtre est déterminée par $(\alpha, \omega) \in \mathbb{R}^+$.
On choisit ces deux paramètres en fonction de nos données.

$$\begin{cases} c = \frac{1-2\cdot e^{-\alpha}\cdot cos(\omega)+e^{-2\cdot\alpha}}{e^{-\alpha}\cdot sin(\omega)} \\ c_1 = \frac{k\cdot\alpha}{\alpha^2+\omega^2} \\ c_2 = \frac{k\cdot\omega}{\alpha^2+\omega^2} \\ b_1 = -2\cdot e^{-\alpha}\cdot cos(\omega) \\ b_2 = e^{-2\cdot\alpha} \end{cases} \qquad \begin{cases} a = -c\cdot e^{-\alpha}\cdot sin(\omega) \\ a_0 = c_2 \\ a_1 = (-c_2\cdot cos(\omega) + c_1\cdot sin(\omega))\cdot e^{-\alpha} \\ a_2 = a_1 - c_2\cdot b_1 \\ a_3 = -c_2\cdot b_2 \end{cases}$$

$$k = \frac{(1-2\cdot e^{-\alpha}\cdot cos(\omega)+e^{-2\cdot\alpha})\cdot(\alpha^2+\omega^2)}{2\cdot\alpha\cdot e^{-\alpha}\cdot sin(\omega)+\omega-\omega\cdot e^{-2\cdot\alpha}}$$

Cas bi-dimensionnel

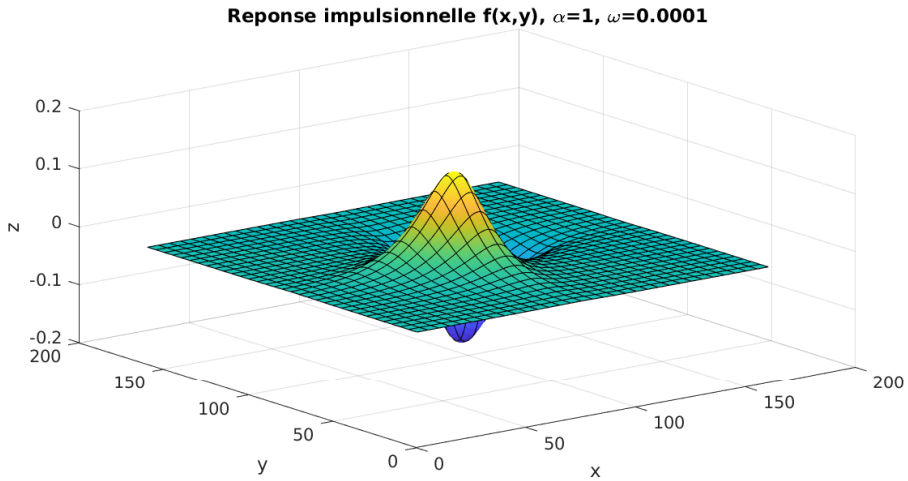
Réponse impulsionnelle $f(x, y)$

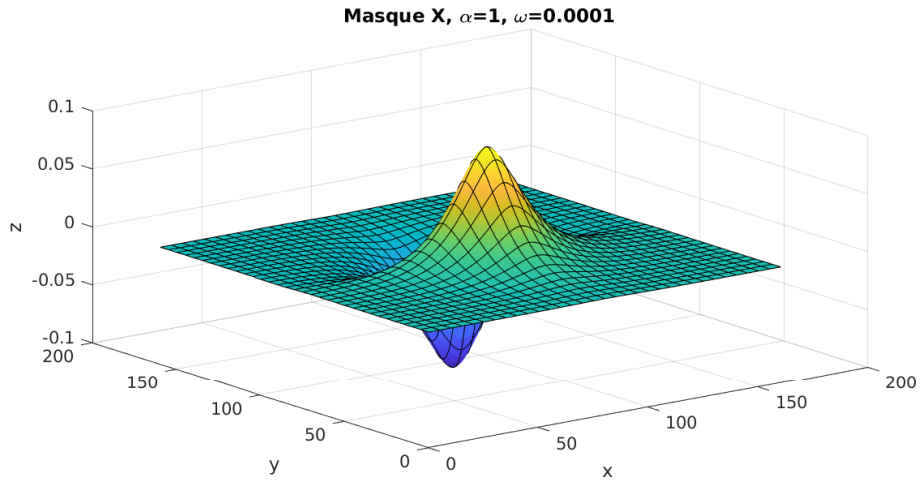
$$f(x,y) = -\frac{c \cdot k}{\alpha^2 + \omega^2} \cdot e^{-\alpha \cdot (|x|+|y|)} \cdot [sin(\omega \cdot x) \cdot (\alpha \cdot sin(\omega \cdot |y|) + \omega \cdot cos(\omega \cdot |y|)) + sin(\omega \cdot y) \cdot (\alpha \cdot sin(\omega \cdot |x|) + \omega \cdot cos(\omega \cdot |x|))]$$

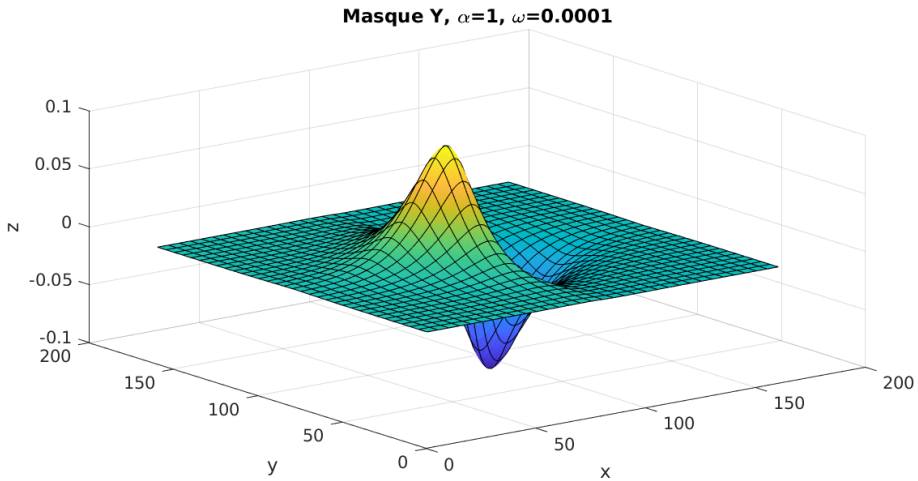
$$f(x,y) = \underbrace{f(x) \cdot h(y)}_{X} + \underbrace{h(x) \cdot f(y)}_{Y} \qquad \text{avec } (x,y) \in \mathbb{R}^2$$

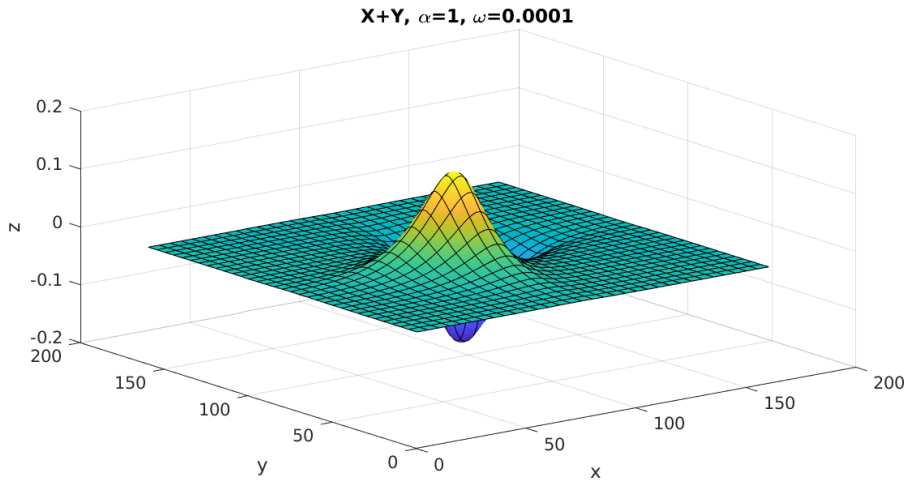
Remarque

Le filtre est donc séparable en deux masques X et Y.


Reponse impulsionnelle f(x,y), α=1, ω=0.0001
z
0.2
0.1
0
-0.1
-0.2
200
150
100
50
0
y
0
50
100
150
200
x


Masque X, α=1, ω=0.0001
z
0.1
0.05
0
-0.05
-0.1
200
150
100
50
0
y
0
50
100
150
200
x


Masque Y, α=1, ω=0.0001
z
0.1
0.05
0
-0.05
-0.1
200
150
100
50
0
y
0
50
100
150
200
x


X+Y, α=1, ω=0.0001
z
0.2
0.1
0
-0.1
-0.2
200
150
100
50
0
y
0
50
100
150
200
x

Sommaire

Etape 1/4 : Gradients selon x et y et lissage (**récursif**)

Gradient selon x

$$Y^+(i,j) = I(i,j-1) - b_1 \cdot Y^+(i,j-1) - b_2 \cdot Y^+(i,j-2)$$
$$j = 1,...,NCL \quad i = 1,...,NLG$$

$$Y^-(i,j) = I(i,j+1) - b_1 \cdot Y^-(i,j+1) - b_2 \cdot Y^-(i,j+2)$$
$$j = NCL,...,1 \quad i = 1,...,NLG$$

$$S(i,j) = a \cdot (Y^+(i,j) - Y^-(i,j)) \quad j = 1,...,NCL \quad i = 1,...,NLG$$

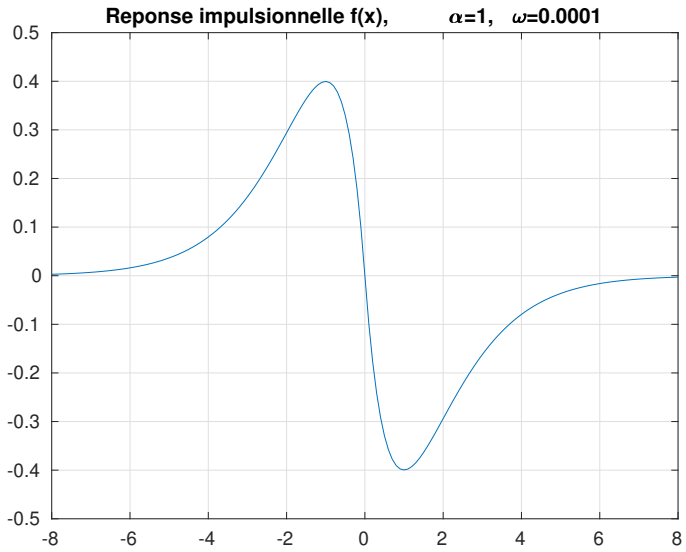
Lissage

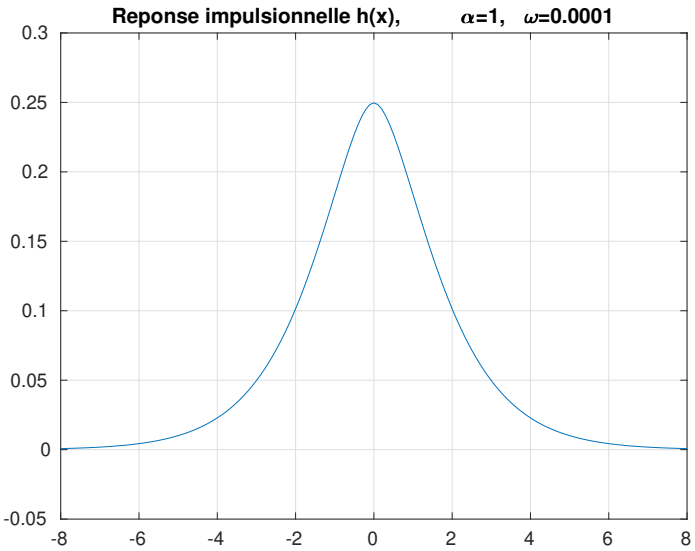
$$R^+(i,j) = a_0 \cdot S(i,j) + a_1 \cdot S(i-1,j) - b_1 \cdot R^+(i-1,j) - b_2 \cdot R^+(i-2,j)$$
$$i = 1,...,NLG \quad j = 1,...,NCL$$

$$R^-(i,j) = a_2 \cdot S(i+1,j) + a_3 \cdot S(i+2,j) - b_1 \cdot R^-(i+1,j) - b_2 \cdot R^-(i+2,j)$$
$$i = NLG,...,1 \quad j = 1,...,NCL$$

$$R(i,j) = R^-(i,j) + R^+(i,j) \quad i = 1,...,NLG \quad j = 1,...,NCL$$

Etape 1/4 : Gradient selon x et lissage (**convolutions**)





$f(n)$ (gradient selon x)

$$\mathcal{I} = \begin{pmatrix} i & i & i & i & i & i \\ i & i & i & i & i & i \\ i & i & i & i & i & i \\ i & i & i & i & i & i \\ i & i & i & i & i & i \\ i & i & i & i & i & i \end{pmatrix}$$

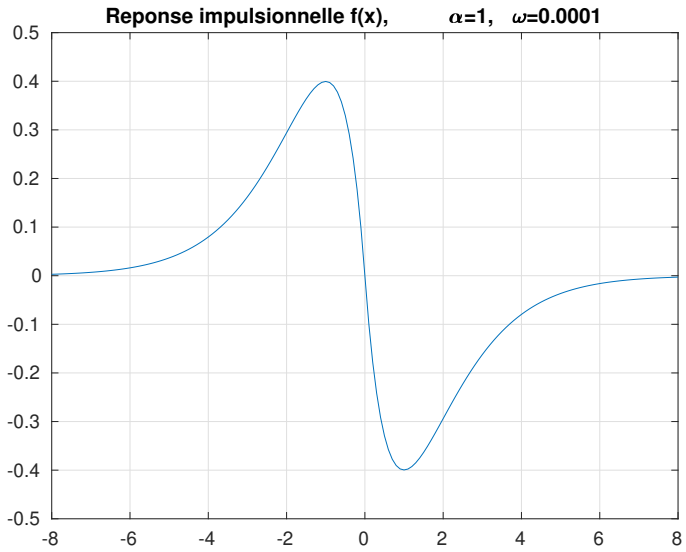
$h(n)$ (lissage)

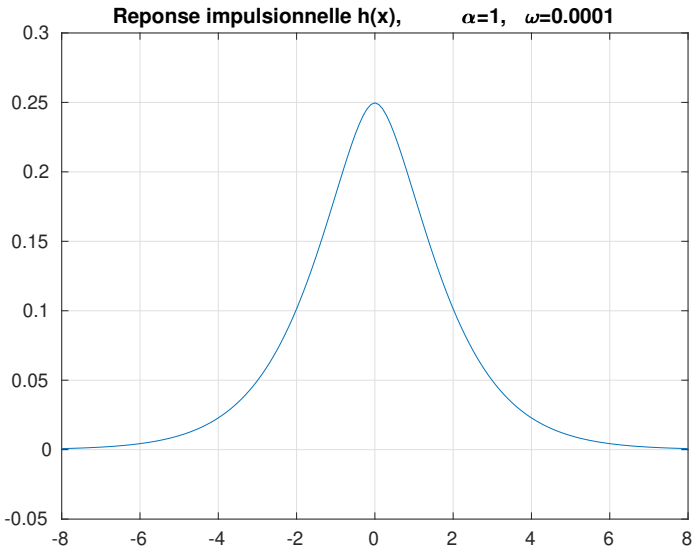
$n \in [\![-N \cdot P; N \cdot P]\!]$

N : nombre de lignes dans l'image

P : nombre de colonnes dans l'image

Etape 1/4 : Gradient selon y et lissage (**convolutions**)





$$\mathcal{I} = \begin{pmatrix} i & i & i & i & i & i \\ i & i & i & i & i & i \\ i & i & i & i & i & i \\ i & i & i & i & i & i \\ i & i & i & i & i & i \\ i & i & i & i & i & i \end{pmatrix} \xrightarrow{h(n)\ \text{(lissage)}} \Big\downarrow f(n)\ \text{(gradient selon y)}$$

$n \in [\![-N \cdot P; N \cdot P]\!]$

N : nombre de lignes dans l'image

P : nombre de colonnes dans l'image

Etape 2/4 : Norme du gradient et direction du gradient

Norme du gradient

$$A(m,n) = \sqrt{r(m,n)^2 + s(m,n)^2)}$$

$$\text{avec} \begin{cases} r \text{ gradient selon x} \\ s \text{ gradient selon y} \end{cases}$$

Direction du gradient

$$arg = arctan(\frac{s(m,n)}{r(m,n)})$$

Sommaire

Etape 3/4 : Suppression des non-maxima locaux

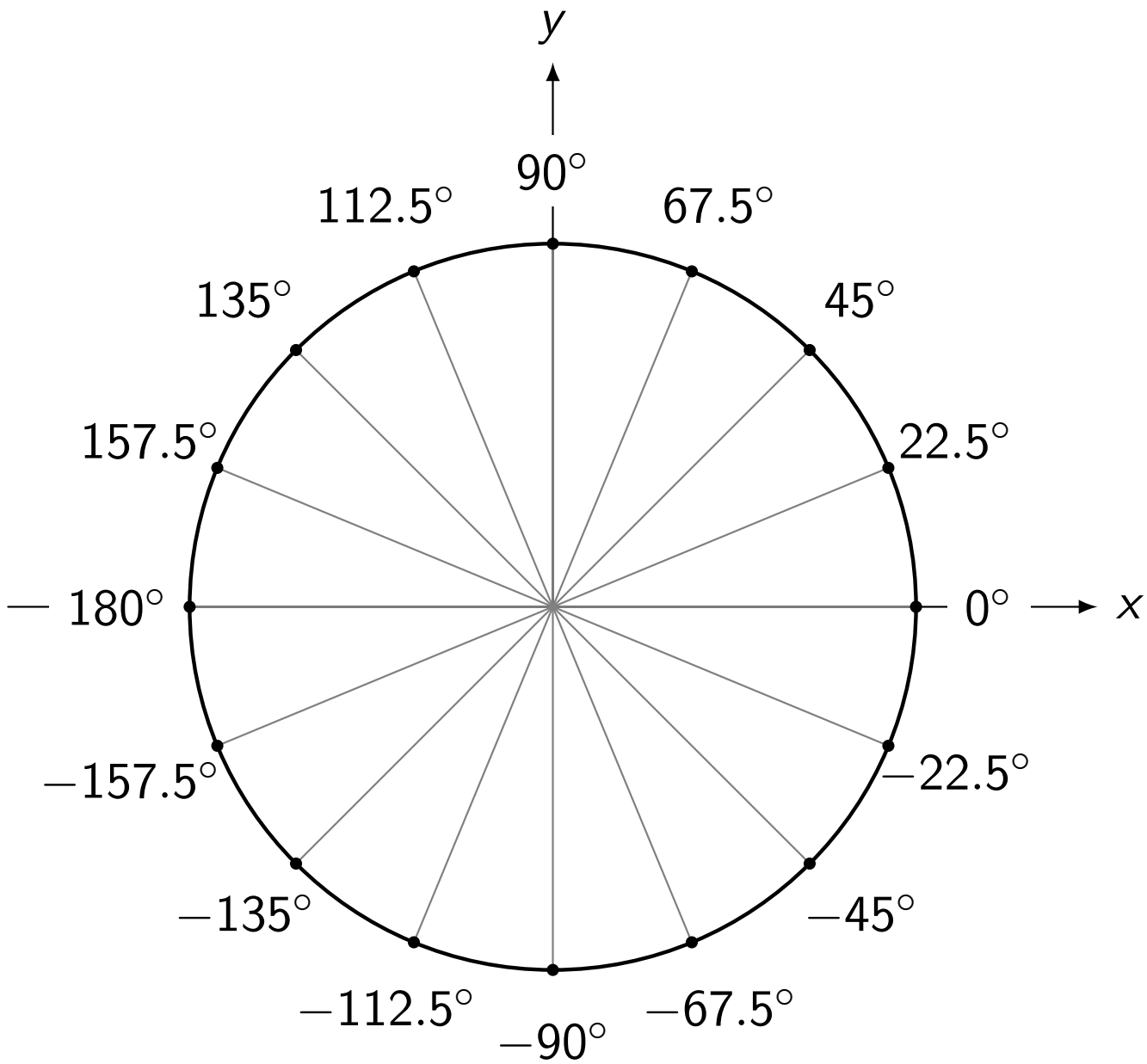
Définition

Méthode appliquée sur la norme du gradient qui permet d'affiner les contours

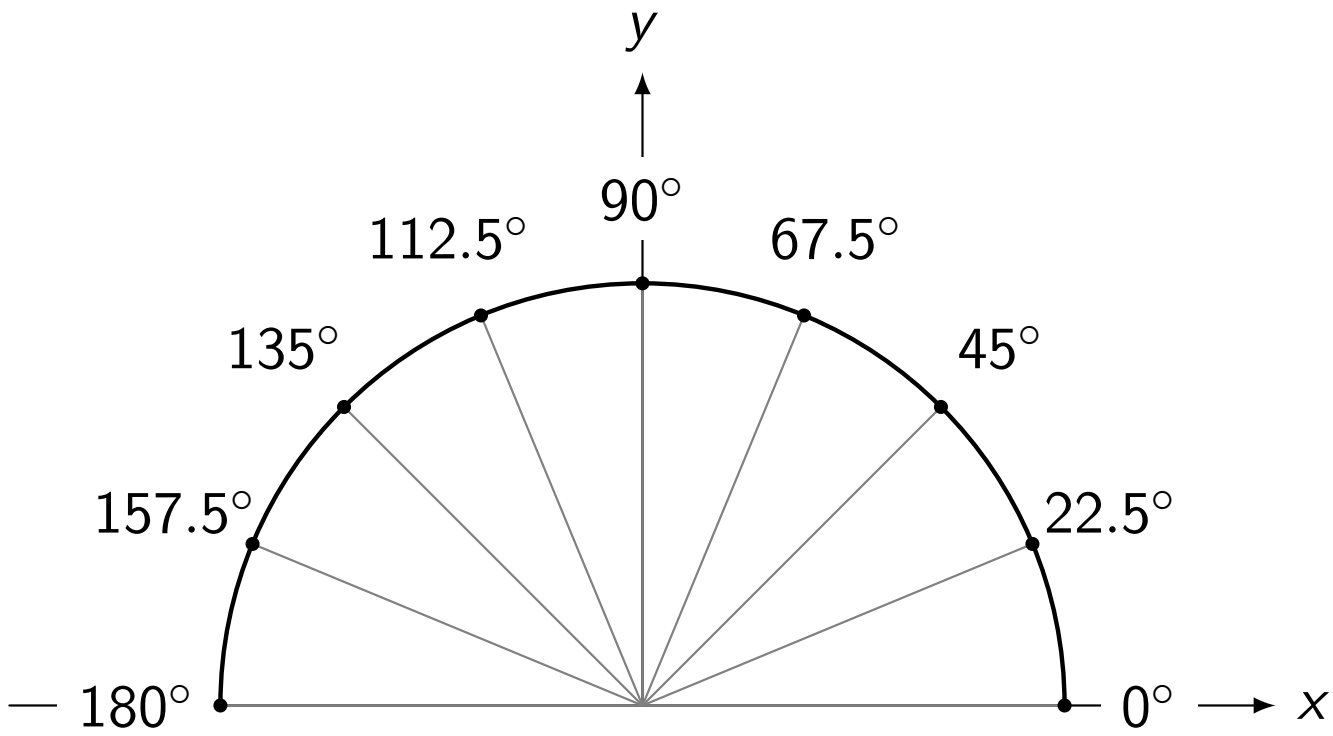
Méthode

Mettre à 0 les pixels qui ne constituent pas un contour.
Un pixel de la norme du gradient est considéré comme contour s'il forme un maximum local dans la direction du gradient.

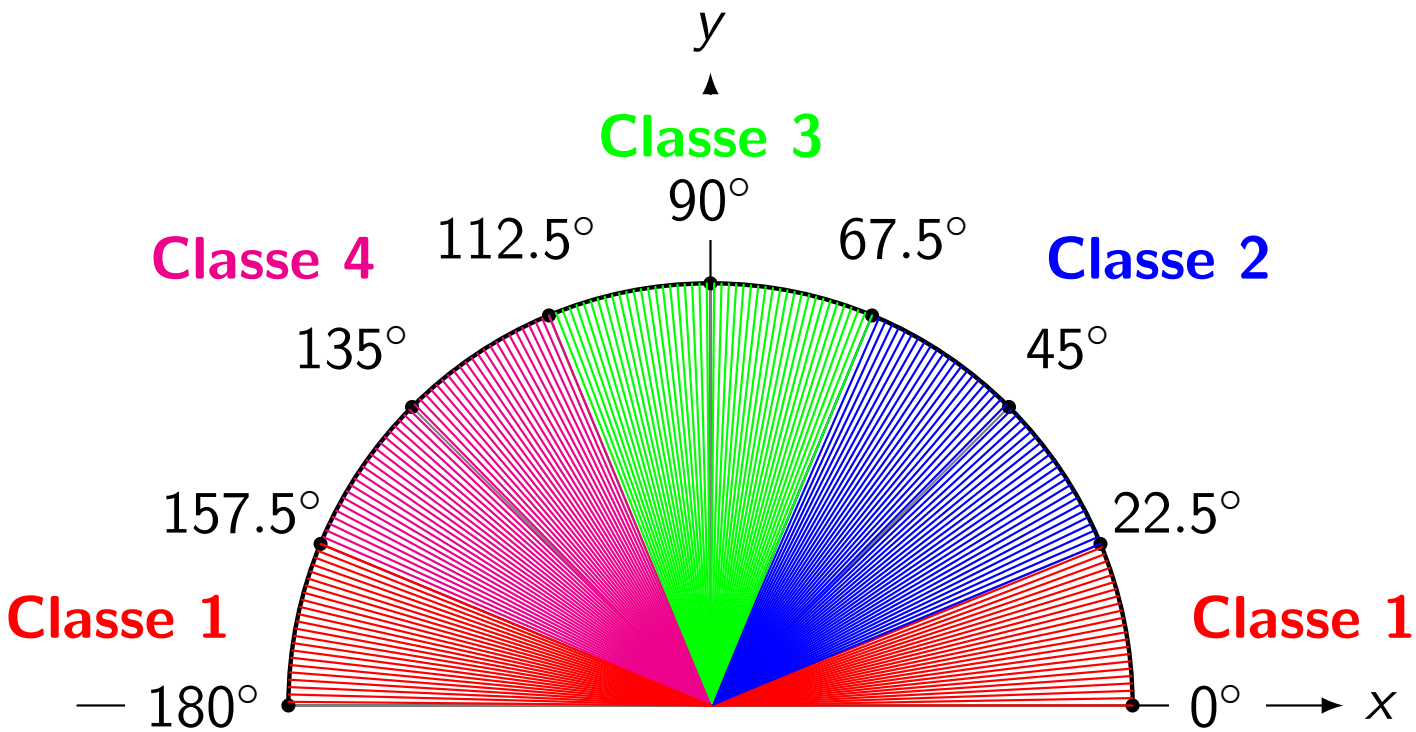
Etape 3/4 : Suppression des non-maxima locaux



Etape 3/4 : Suppression des non-maxima locaux



Etape 3/4 : Suppression des non-maxima locaux



Etape 3/4 : Suppression des non-maxima locaux

Direction du gradient

$$\begin{pmatrix} D & D & D & D & D & D \\ D & D & D & D & D & D \\ D & D & D & D & D & D \\ D & D & D & D & D & D \\ D & D & D & D & D & D \\ D & D & D & D & D & D \end{pmatrix}$$

Norme du gradient

$$\begin{pmatrix} N & N & N & N & N & N \\ N & N & N & N & N & N \\ N & N & N & N & N & N \\ N & N & N & N & N & N \\ N & N & N & N & N & N \\ N & N & N & N & N & N \end{pmatrix}$$

Etape 3/4 : Suppression des non-maxima locaux

Classe 1 : $arg \in [0; 22.5[$ ou $arg \in [157.5; 180]$

Direction du gradient

$$\begin{pmatrix} D & D & D & D & D & D \\ D & \mathbf{D} & D & D & D & D \\ D & D & D & D & D & D \\ D & D & D & D & D & D \\ D & D & D & D & D & D \\ D & D & D & D & D & D \end{pmatrix}$$

Norme du gradient

$$\begin{pmatrix} N & N & N & N & N & N \\ \mathbf{N} & \mathbf{N} & \mathbf{N} & N & N & N \\ N & N & N & N & N & N \\ N & N & N & N & N & N \\ N & N & N & N & N & N \\ N & N & N & N & N & N \end{pmatrix}$$

Si $non(\mathbf{N} \geq max(\mathbf{N}, \mathbf{N}))$ alors $\mathbf{N} \leftarrow 0$

Etape 3/4 : Suppression des non-maxima locaux

Classe 2 : $arg \in [22.5; 67.5[$

Direction du gradient

$$\begin{pmatrix} D & D & D & D & D & D \\ D & \mathbf{D} & D & D & D & D \\ D & D & D & D & D & D \\ D & D & D & D & D & D \\ D & D & D & D & D & D \\ D & D & D & D & D & D \end{pmatrix}$$

Norme du gradient

$$\begin{pmatrix} N & N & \mathbf{N} & N & N & N \\ N & \mathbf{N} & N & N & N & N \\ \mathbf{N} & N & N & N & N & N \\ N & N & N & N & N & N \\ N & N & N & N & N & N \\ N & N & N & N & N & N \end{pmatrix}$$

Si $non(\mathbf{N} \geq max(\mathbf{N}, \mathbf{N}))$ alors $\mathbf{N} \leftarrow 0$

Etape 3/4 : Suppression des non-maxima locaux

Classe 3 : $arg \in [67.5; 112.5[$

Direction du gradient

$$\begin{pmatrix} D & D & D & D & D & D \\ D & \mathbf{D} & D & D & D & D \\ D & D & D & D & D & D \\ D & D & D & D & D & D \\ D & D & D & D & D & D \\ D & D & D & D & D & D \end{pmatrix}$$

Norme du gradient

$$\begin{pmatrix} N & \mathbf{N} & N & N & N & N \\ N & \mathbf{N} & N & N & N & N \\ N & \mathbf{N} & N & N & N & N \\ N & N & N & N & N & N \\ N & N & N & N & N & N \\ N & N & N & N & N & N \end{pmatrix}$$

Si $non(\mathbf{N} \geq max(\mathbf{N}, \mathbf{N}))$ alors $\mathbf{N} \leftarrow 0$

Etape 3/4 : Suppression des non-maxima locaux

Classe 4 : $arg \in [112.5; 157.5[$

Direction du gradient

$$\begin{pmatrix} D & D & D & D & D & D \\ D & \mathbf{D} & D & D & D & D \\ D & D & D & D & D & D \\ D & D & D & D & D & D \\ D & D & D & D & D & D \\ D & D & D & D & D & D \end{pmatrix}$$

Norme du gradient

$$\begin{pmatrix} \mathbf{N} & N & N & N & N & N \\ N & \mathbf{N} & N & N & N & N \\ N & N & \mathbf{N} & N & N & N \\ N & N & N & N & N & N \\ N & N & N & N & N & N \\ N & N & N & N & N & N \end{pmatrix}$$

Si $non(\mathbf{N} \geq max(\mathbf{N}, \mathbf{N}))$ alors $\mathbf{N} \leftarrow 0$

Etape 3/4 : Suppression des non-maxima locaux

Remarque

Après cette étape, l'image n'est pas encore binarisée mais on a mis à 0 les pixels qui ne constituaient pas un contour.

Sommaire

Etape 4/4 : Seuillage par hystérésis

Définition

Le seuillage par hystérésis permet de retirer les faux contours.

Méthode

Pour cela, on définit deux seuils :

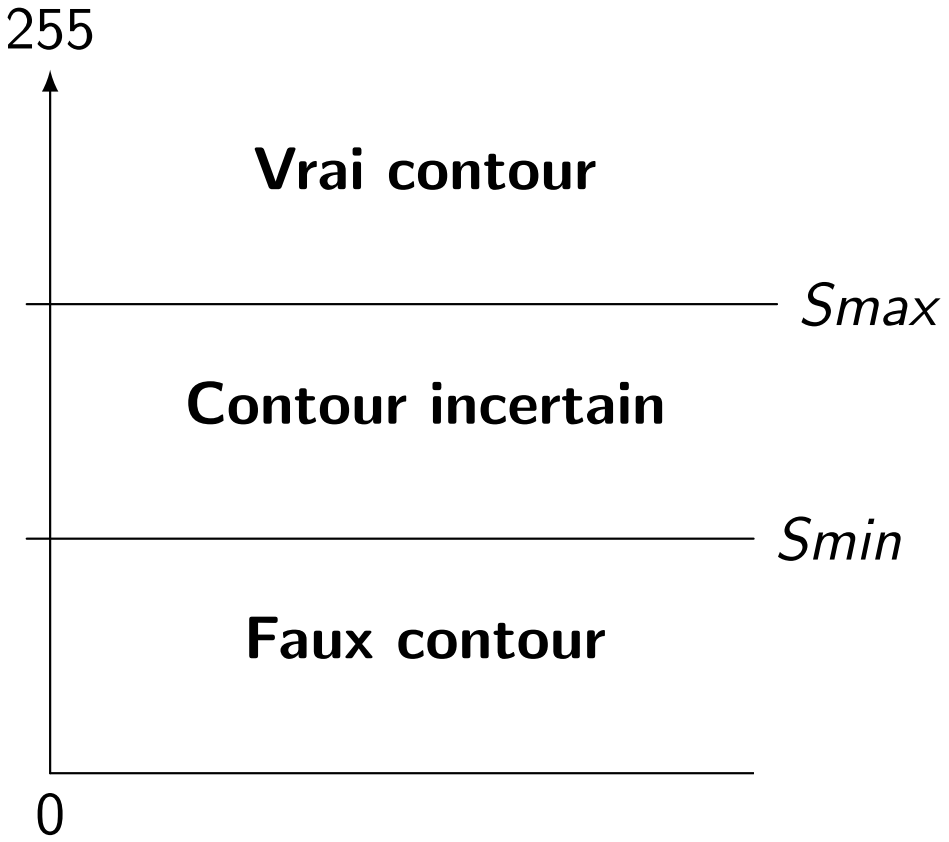
- *Smin* : seuil minimum
- *Smax* : seuil maximum

Pour chaque pixel de la norme du gradient obtenu après la suppression des non-maxima locaux, on regarde sa valeur.

Etape 4/4 : Seuillage par hystérésis

Norme du gradient après suppression des non-maxima locaux

$$\begin{pmatrix} N & N & N & N & N & N \\ N & \mathbf{N} & N & N & 0 & N \\ N & N & N & 0 & N & N \\ N & N & N & N & N & N \\ N & 0 & N & 0 & N & N \\ N & N & N & N & 0 & N \end{pmatrix}$$



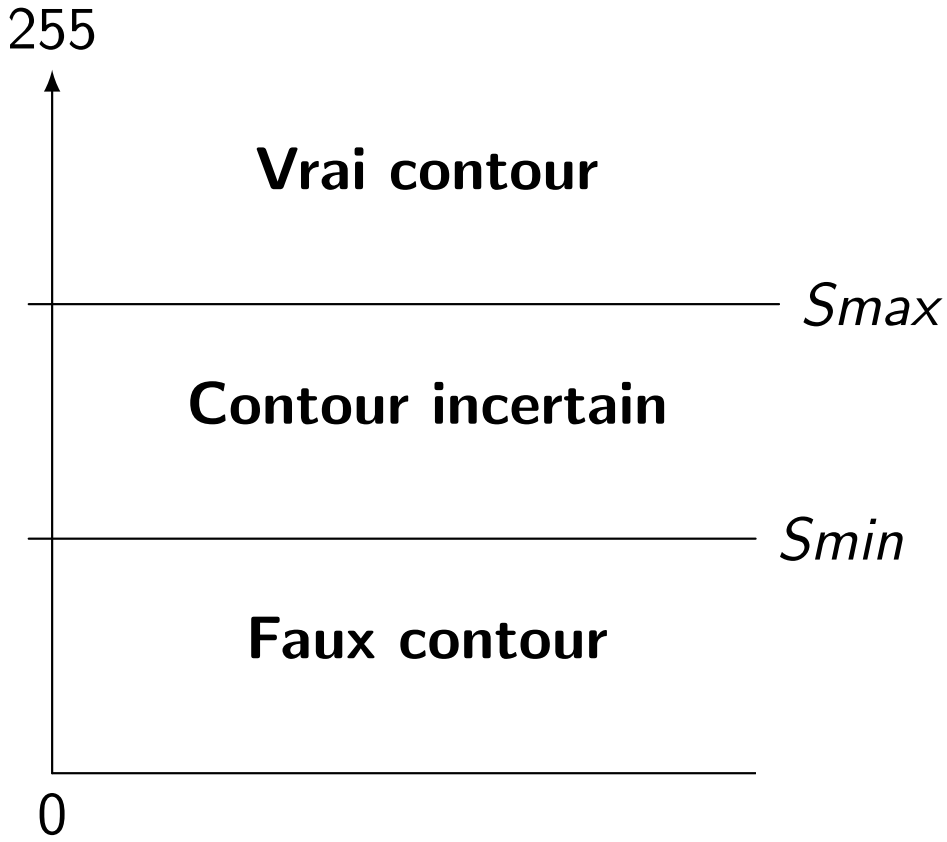
Remarque

L'image obtenue est maintenant binarisée.
Elle contient les contours de l'image initiale et un peu de bruit.

Etape 4/4 : Seuillage par hystérésis

Norme du gradient après suppression des non-maxima locaux

$$\begin{pmatrix} \mathbf{N} & \mathbf{N} & \mathbf{N} & N & N & N \\ \mathbf{N} & \mathbf{N} & \mathbf{N} & N & 0 & N \\ \mathbf{N} & \mathbf{N} & \mathbf{N} & 0 & N & N \\ N & N & N & N & N & N \\ N & 0 & N & 0 & N & N \\ N & N & N & N & 0 & N \end{pmatrix}$$

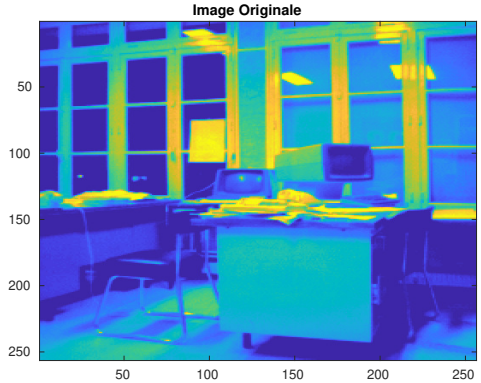


Remarque

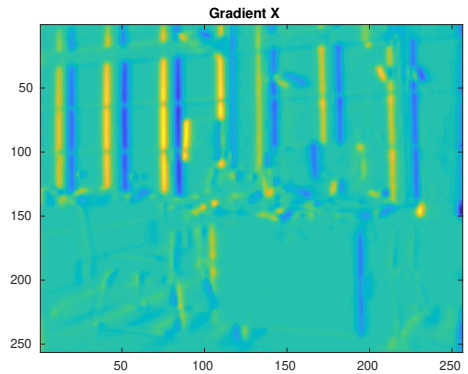
L'image obtenue est maintenant binarisée.
Elle contient les contours de l'image initiale et un peu de bruit.

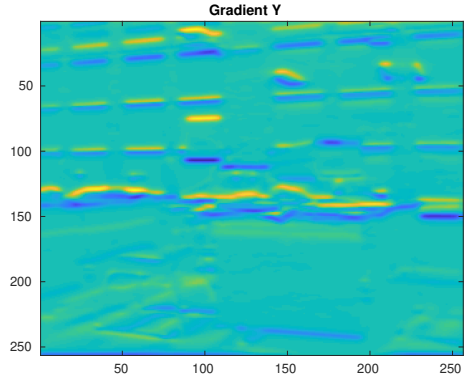
Sommaire

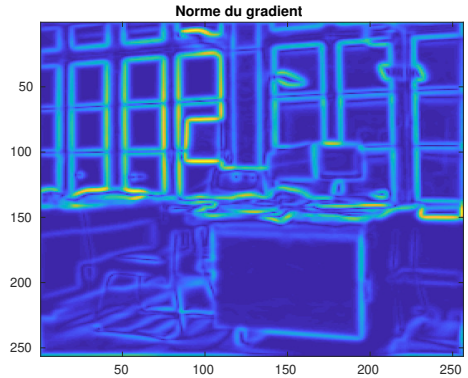
Résultats

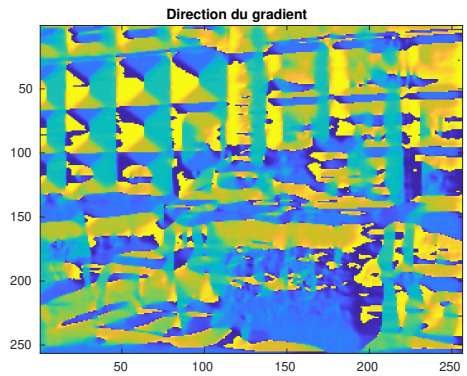


Récursif

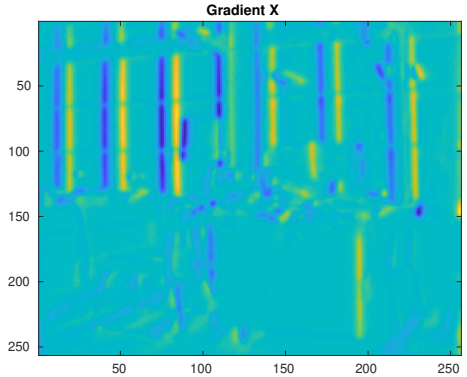


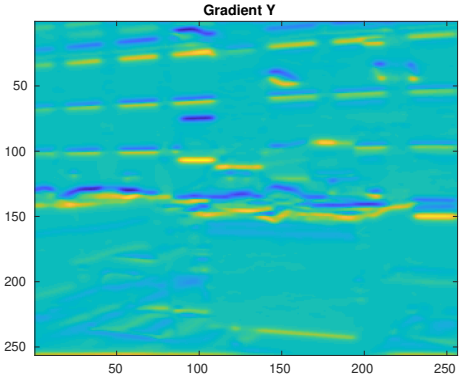


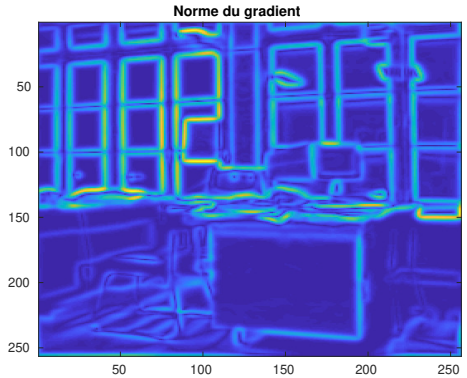


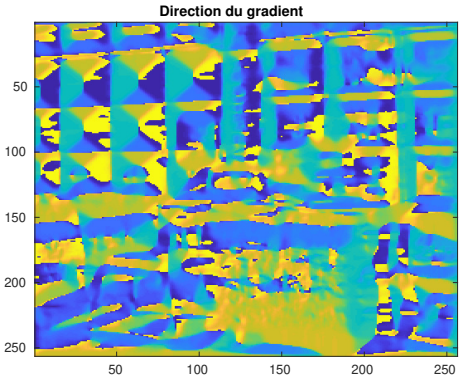


Convolutions

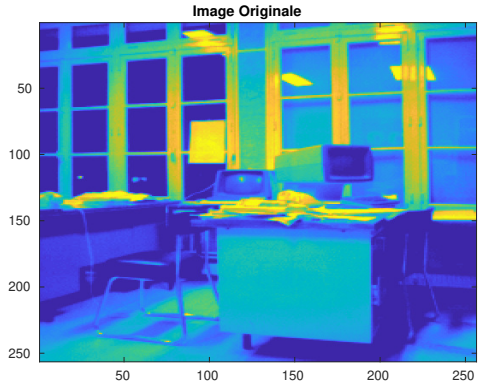




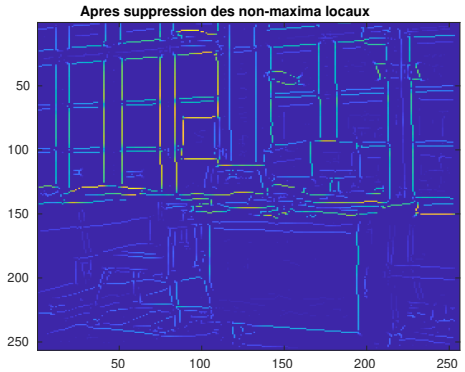


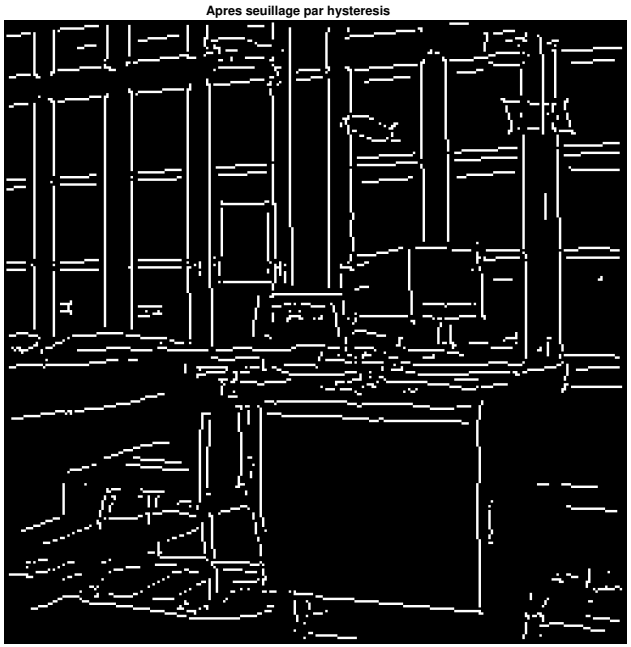


Résultats

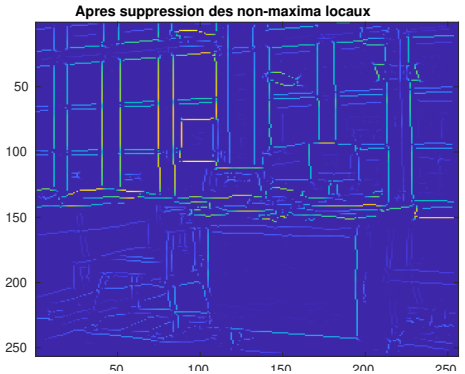


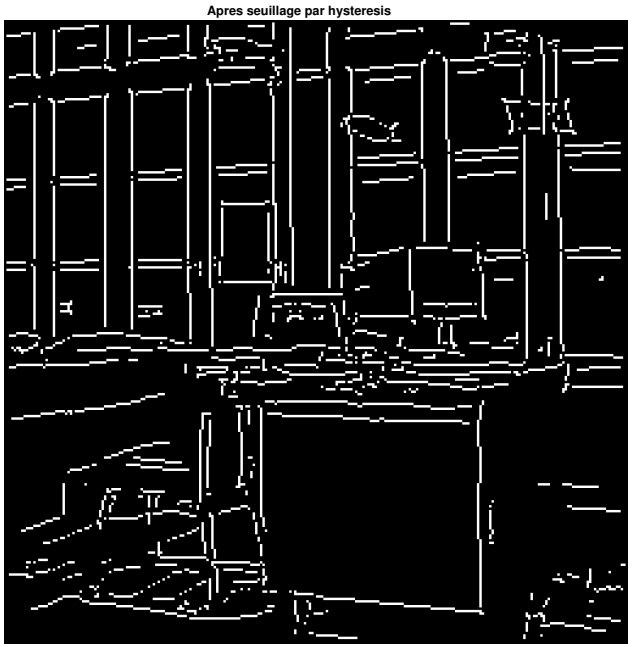
Récursif



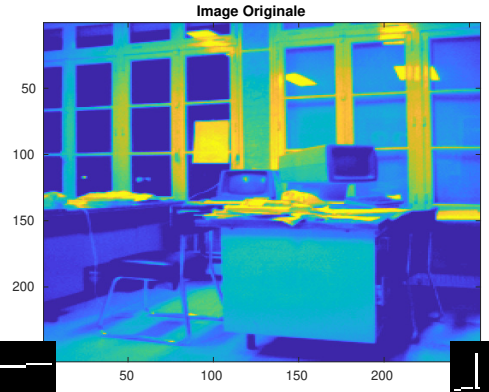


Convolutions





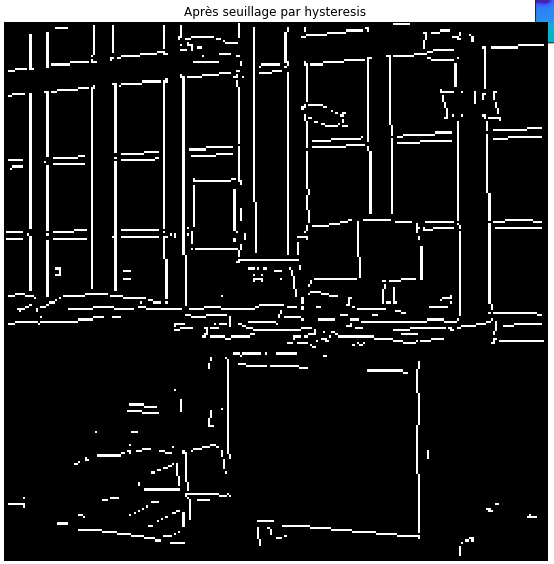
Résultats

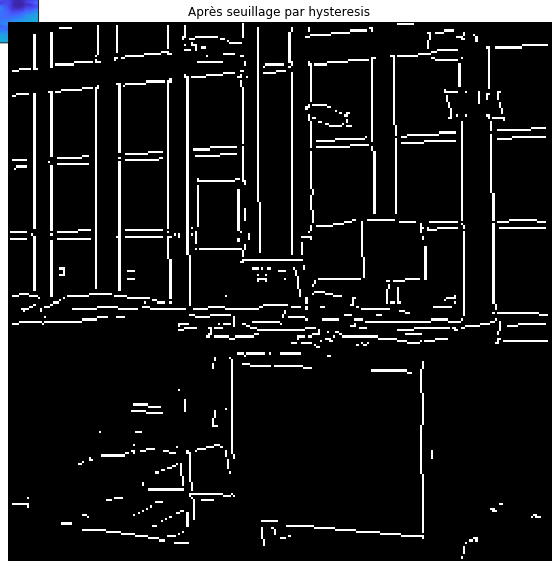


Récursif

Convolutions

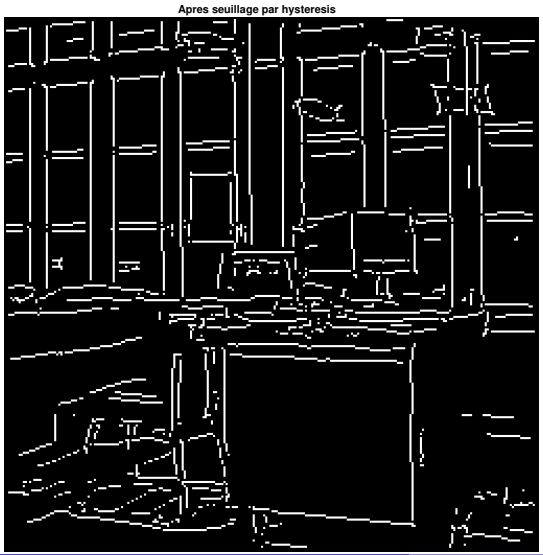
Python

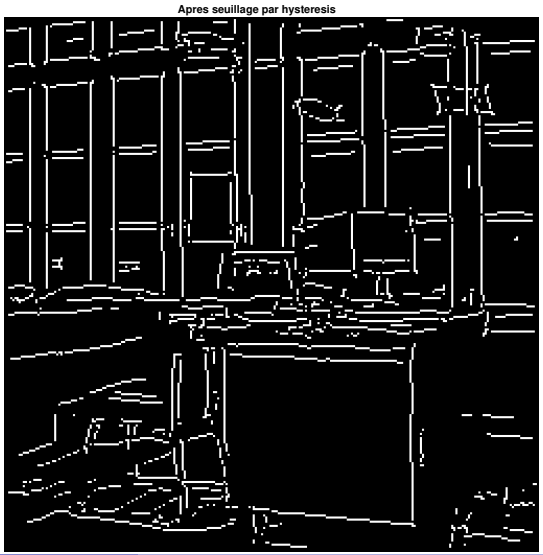




Python

Matlab





Matlab

Sommaire

Sommaire

Bibliographie

- **Article**

  Titre : Using Canny's Criteria to Derive a Recursively Implemented Optimal Edge Detector

  Auteur : RACHID DERICHE

- Deriche edge detector
  https://en.wikipedia.org/wiki/Deriche_edge_detector
- Can't get clean output in my MATLAB implementation of Canny-Deriche
  https://stackoverflow.com/questions/14132356/cant-get-clean-output-in-my-matlab-implementation-of-cann
- Canny edge detector
  https://en.wikipedia.org/wiki/Canny_edge_detector
- Canny Edge Detection Step by step
  https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html

Bibliographie

- Détection de contours : les opérateurs de Canny-Deriche
  https://perso.esiee.fr/~coupriem/Algo/algoima.html
- Extraction de contours et son extension du contour actif
  http://ninebill.free.fr/ExtractionContours/detection/canny.html
- JAVA demo of Canny/Deriche-like filter (EPFL Biomedical Imaging Group)
  http://bigwww.epfl.ch/demo/ip/demos/edgeDetector/
- Segmentator – Fix gramag export error with deriche filter (3D python)
  https://github.com/ofgulban/segmentator
- Edge Detection with MATLAB
  https://fr.mathworks.com/videos/edge-detection-with-matlab-119353.html
- Gaël Deest (pdf)
  www.theses.fr/2017REN1S102/abes

Bibliographie

- Canny en python (OpenCV)

  https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html
- Convolution in Two Dimensions

  http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MARBLE/low/space/convol.htm
- Convolution en 2D

  http://web.stanford.edu/group/sequoia/cgi-bin/node/185
- EECS 442 – Computer vision

  http://vhosts.eecs.umich.edu/vision//teaching/EECS442_2012/lectures/lecture14.pdf
- DETECTION DE CONTOURS

  http://www.lirmm.fr/~strauss/PageImage3/EdgeDetection.pdf

Bibliographie

- Techniques d'extraction de contours
  ftp://ftp-sop.inria.fr/athena/Team/Rachid.Deriche/Lectures/Master-Stic-IGMMV/techniques_contours.pdf
- Détection de contours
  http://devernay.free.fr/cours/vision/pdf/c3.pdf
- Bases du traitement des images - Détection de contours
  http://webia.lip6.fr/~thomen/Teaching/BIMA/cours/contours.pdf
- La détection decontours dans des images à niveaux de gris : mise en œuvre et sélection de détecteurs
  http://docnum.univ-lorraine.fr/public/INPL_T_1991_ZIOU_D.pdf
- Detection de contours
  http://dept-info.labri.fr/~achille/enseignement/TI/TI-ch4-notes.pdf
- Segmentation d'image : Contours
  http://www.lgi2p.mines-ales.fr/~montesin/CoursPDF/