TRAITEMENT D'IMAGES - DÉTECTION DE CONTOURS

Nicolas CHIREUX

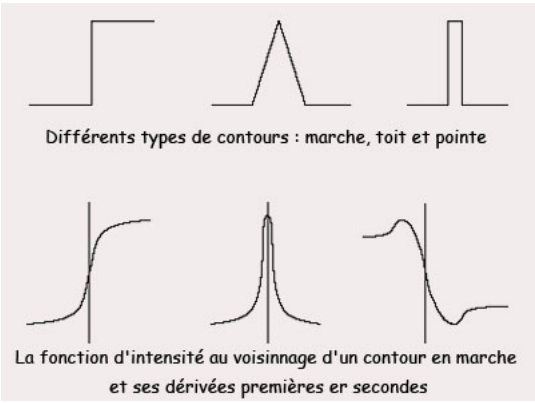
La détection de contours est un domaine du traitement d'image très riche. De très nombreuses méthodes ont été développées depuis les années 1970 : elles présentent toutes des avantages et des inconvénients et le choix d'une méthode sera le plus souvent guidé par le résultat obtenu par rapport à l'usage souhaité (création de filtres, traitement d'image, reconnaissance de formes...)

1 Introduction à la détection de contours

1.1 Définition

Très schématiquement, les contours sont les lieux de variations de l'intensité en niveaux de gris rapides dans une direction et lentes dans la direction perpendiculaire (il y a très peu de travaux sur les contours dans les images couleurs).

Dans cette approche, on suppose que l'image est une mosaïque de régions parfaitement homogènes. C'est à dire que les contours recherchés sont de type créneaux. De plus, la transition étant stricte, un contour doit être une chaîne de pixels d'épaisseur 1. Cette restriction sur la nature du contour a été imposée dans un premier temps pour des raisons de formalisation mathématique.



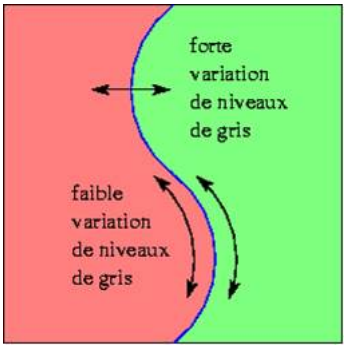


FIGURE 1 – Types de contours

Dans toute la suite nous travaillerons sur des images en niveau de gris.

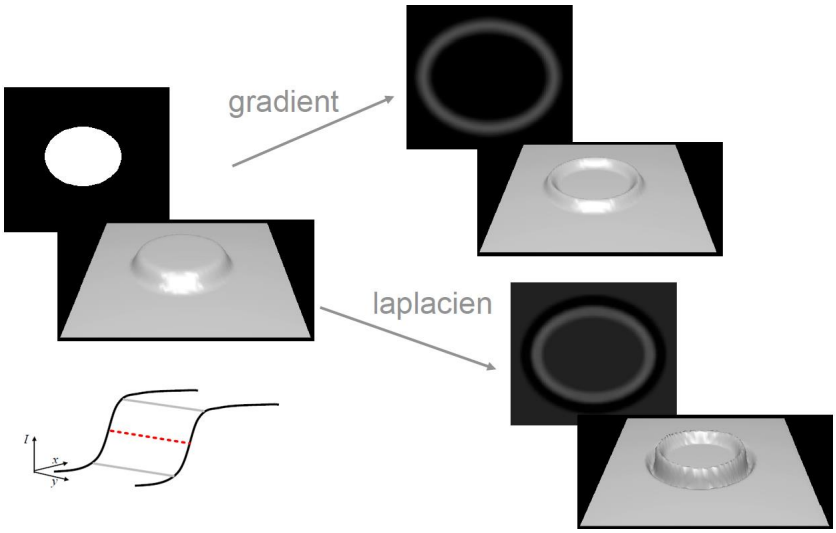
1.2 Les catégories de méthodes

1.2.1 Les méthodes dérivatives

La notion de contour étant reliée à celle de variation, il va falloir évaluer la variation du niveau de gris en chaque pixel. Les méthodes dérivatives s'appuient sur la constatation que les contours d'image sont traduits généralement par les transitions rapides de l'image, et que les variations lentes seront éliminées par dérivation. Nous procèderons donc :

— soit par recherche d'un maximum local du gradient : les principaux algorithmes connus sont Sobel, Prewitt, Kirsh, Canny, Dériche...

— soit par recherche du passage par zéro de la dérivée seconde : il existe moins de travaux sur la formalisation de cette méthode qui pourtant donne souvent des résultats de meilleure qualité visuelle.



Dans toute la suite nous travaillerons sur des images en niveau de gris.

FIGURE 2 – Méthodes dérivatives

1.2.2 Les méthodes de contours actifs

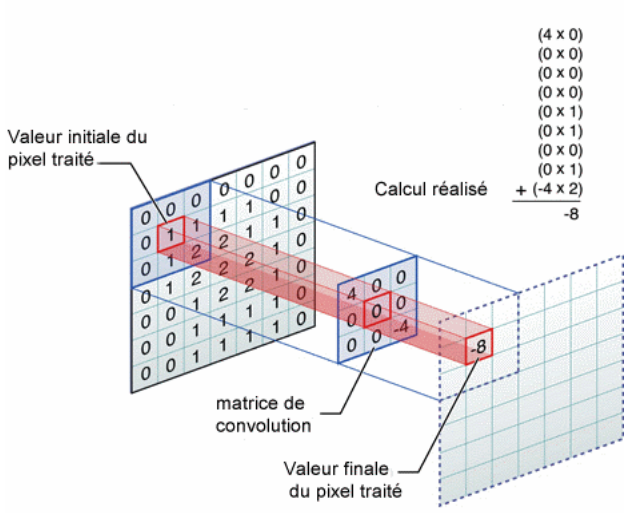
On trouve dans cette catégories les snakes, la méthode des ballons, les contours actifs géodésiques, les level sets...

Nous ne donnerons ici que aperçu succinct des snakes. Les contours actifs (ou snakes) ont été introduits par Kass et Witkin à la fin des années 80. C'est une méthode semi-interactive dont le principe consiste à placer dans l'image au voisinage de la forme à détecter un contour initial. On va faire ensuite évoluer le snake de manière à minimiser son énergie.

2 Les méthodes dérivatives

2.1 Filtres de gradient

2.1.1 Rappels sur le filtrage linéaire



Le filtrage linéaire consiste simplement à remplacer chaque niveau de gris par une combinaison linéaire des niveaux de gris des points voisins.

FIGURE 3 – Méthodes dérivatives

Plus formellement, filtrer signifie convoluer une image I(x,y) avec une fonction f(x,y) appelée réponse impulsionnelle (ou opérateur de convolution) du filtre. Dans le cas continu, l'image filtrée I_f par le filtre f est donnée par :

$$I_f(x,y) = (f * I)(x,y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x',y').I(x-x',y-y')dx'dy' \tag{1}$$

Dans le cas discret, si on applique un filtre représenté par une matrice carrée de dimension n, on aura pour le pixel (i,j) :

$$I_f(i,j) = (f * I)(i,j) = \sum_{k=-n/2}^{n/2} \sum_{l=-n/2}^{n/2} f(k,l).I(i-k,j-l) \tag{2}$$

Il est bien sûr nécessaire de normaliser le résultat. De plus, le résultat peut être négatif, il convient alors d'y ajouter une constante (souvent 128 puisque nous travaillons en niveaux de gris).

2.1.2 Les filtres de gradient

Ce sont des filtres de dérivée première, dits étroits, dont on repère le maximum de leur réponse. Ils se calculent au minimum sur 3 points.

Le gradient, en un pixel d'une image numérique, est un vecteur caractérisé par son amplitude et sa direction. L'amplitude est directement liée à la quantité de variation locale des niveaux de gris. La direction du gradient est orthogonale à la frontière qui passe au point considéré.

La méthode la plus simple pour estimer un gradient est donc de faire un calcul de variation mono-dimensionnelle i.e. en ayant choisi une direction donnée : en général les deux directions privilégiées sont celles associées au maillage de l'image.

En remarquant que $f'(x) \simeq \dfrac{f(x+1) - f(x-1)}{2}$, nous obtenons les deux noyaux de convolution suivants pour le gradient horizontal GH et vertical GV :

$$GH = \begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \quad \text{et} \quad GV = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \tag{3}$$

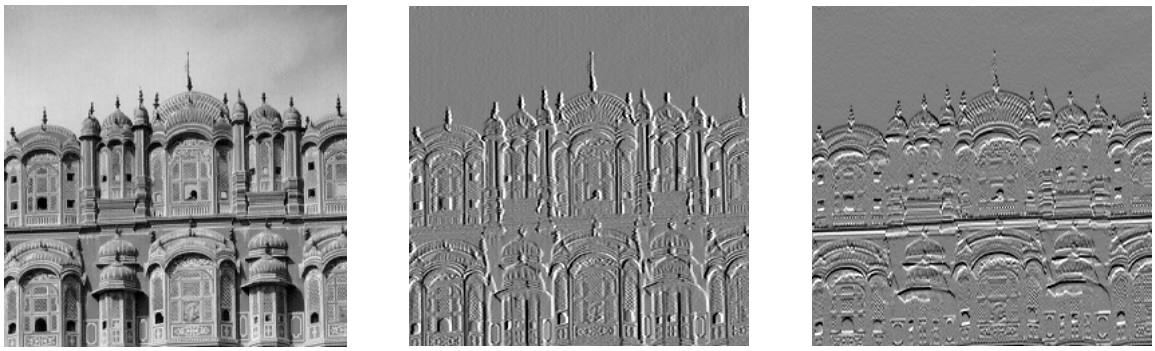


FIGURE 4 – Gradient horizontal et vertical

Les contours horizontaux ressortent mieux sur l'image de gradient vertical. Les contours verticaux ressortent mieux sur l'image de gradient horizontal.

Rem : comme les résultats du gradient normalisé vont de -128 à 128, il faut penser à décaler toutes les valeurs de la matrice avec un shift de 128.

La dérivation accentuant le bruit (pixels parasites de répartition aléatoire), des filtres dérivés, plus robustes, on été proposés. Ces filtres sont certes moins précis que le filtre gradient "pur" mais sont plus fiables de par leur robustesse. Il reste néanmoins utile de les faire précéder de filtres locaux débruiteurs, tels le filtre médian. Nous nous intéresserons aux filtres classiques suivants :

- **Filtre de Prewitt** : calculé sur 9 points, ce filtre effectue une moyenne locale sur 3 points en même temps que la dérivation. Il est défini par le double masque suivant (normalisé par un facteur de $1/3$)

$$PrewittH = \frac{1}{3}\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad \text{et} \quad PrewittV = \frac{1}{3}\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \tag{4}$$

- **Filtre de Sobel** : calculé aussi sur 9 points, ce filtre est une variante du filtre de Prewitt. Il est défini par le double masque suivant (normalisé par un facteur de $1/4$) qui sur-pondère les directions privilégiées :

$$SobelH = \frac{1}{4}\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \quad \text{et} \quad SobelV = \frac{1}{4}\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \tag{5}$$

- **Filtre de Kirsch** : calculé aussi sur 9 points, ce filtre est aussi une variante du filtre de Prewitt. Il est défini par le double masque suivant (normalisé par un facteur de $1/15$) :

$$KirschH = \frac{1}{15}\begin{pmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{pmatrix} \quad \text{et} \quad KirschV = \frac{1}{15}\begin{pmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{pmatrix} \tag{6}$$

Le filtrage complet renvoie la norme du filtrage de gradient soit en notant f_H et f_V les filtres verticaux et horizontaux :

$$I_f(x,y) = \sqrt{(f_H * I)(x,y)^2 + (f_V * I)(x,y)^2} \tag{7}$$

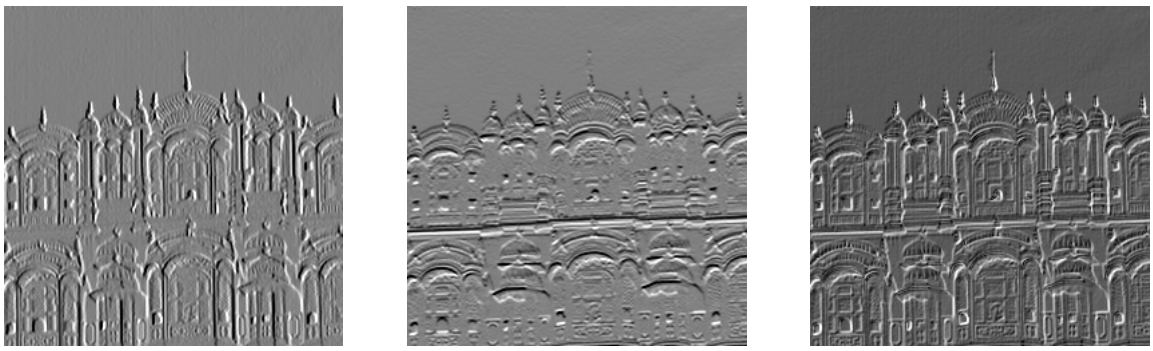


FIGURE 5 – Filtrage de Sobel horizontal, vertical et complet

Les contours obtenus à l'aide des filtres de gradient sont toutefois d'une qualité médiocre et ne peuvent en règle générale être utilisés tels quel car ils sont très souvent bruités, épais, interrompus et non fermés. Certains de ces défauts peuvent être corrigés par des traitements ultérieurs (seuillage...).

L'intérêt des filtres de gradient est la rapidité de mise en oeuvre et le faible coût calculatoire (puisque ce sont des filtres locaux).

2.1.3 Travail informatique

Nous allons reprendre les deux fonctions développées dans le TP précédent permettant de transformer une image en matrice et vice versa. Toutefois nous les adapterons en utilisant les fonctionnalités de numpy afin d'être plus efficaces.

Il est aussi rappelé que le chemin d'accès aux images doit être indiqué comme ci-dessous :

```
import PIL.Image as image
import copy
import numpy as np
from math import *
# vous pouvez indiquer l'arborescence pour atteindre le dossier
link="D :/Nicolas/Informatique/Python/MPstar/Traitement d'image/"
nom="sobelall"
```

La transformation d'une image en matrice utilisera l'attribut **reshape** comme indiqué ci-dessous :

```
def convert(fichier) :
    photo = image.open(fichier)
    taille = photo.size
    pixels = np.array(photo.getdata())
    matrice=np.reshape(pixels,(taille[1],taille[0]))
    return matrice
M=convert(link+nom+".gif")
```

La transformation réciproque utilisera l'attribut **flat** :

```
def deconvert(Mat) :
    ligne = len(Mat)
    colonne = len(Mat[0])
    pixels = list(Mat.flat)
    nouvimage = image.new('L', (colonne,ligne) )
    nouvimage.putdata ( pixels )
    return(nouvimage)
deconvert(M).show()
```

Après avoir entré les deux fonctions ci-dessus, écrire une fonction def **convolution**$(image, noyau, shift)$: à laquelle on fournit une matrice $image$, un noyau de filtrage $noyau$ et un décalage $shift$ permettant de recaler les valeurs de la matrice filtrée entre 0 et 128 et qui retourne la matrice résultat de la convolution de $image$ par $noyau$.

Écrire ensuite une fonction def **Norme**$(image, noyauh, noyauv, shift)$: qui retourne la norme du filtrage choisi. Utiliser ces fonctions pour appliquer à diverses images :

— les gradients horizontal et vertical
— les filtres de Prewitt
— les filtres de Sobel
— les filtres de Kirsch

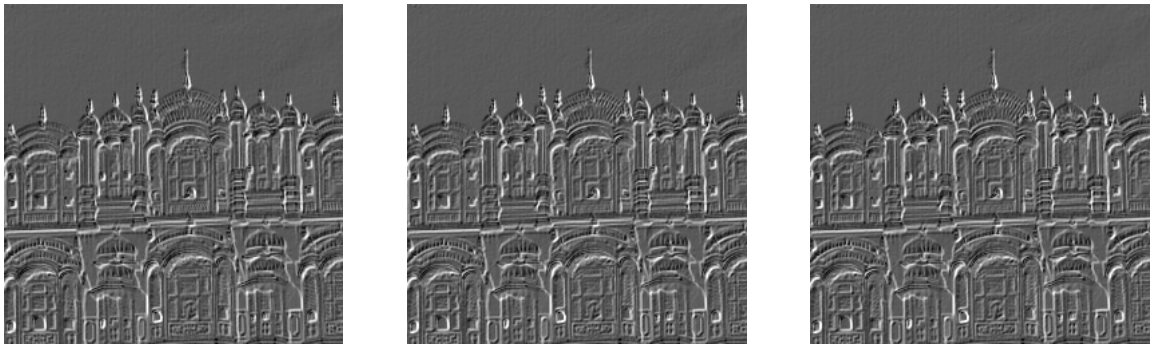


FIGURE 6 – Filtrage de Prewitt, Sobel et Kirsch

2.2 Filtres laplaciens

Ils sont dans la catégorie des filtres larges car étant très sensibles au bruit de l'image, ils devront être précédés de filtres de pré-traitement (floutage gaussien 5x5, passe-bas 10x10 ou 30x30...)

2.2.1 Principe de la méthode

Au lieu de localiser les maxima de la dérivée première, on va localiser les passages par zéro de la dérivée seconde - zero-crossing -.

Étant donné que les images sont à deux dimensions, nous allons travailler sur le laplacien de l'image :

$$\Delta I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \tag{8}$$

Le laplacien peut être approximé par des différences finies comme pour les filtres étroits :

$$\frac{\partial^2 I}{\partial x^2} \to \begin{pmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{pmatrix} \quad \text{et} \quad \frac{\partial^2 I}{\partial y^2} \to \begin{pmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad \text{d'où} \quad \Delta I \to \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{9}$$

Bien que plus sensible au bruit que le gradient, nous verrons que le filtre laplacien respecte mieux les contours fermés.

2.2.2 Nécessité d'un filtrage

Afin de limiter les réponses dues au bruit de l'image I, elle est préalablement filtrée par un filtre G. On obtient alors, grâce aux propriétés de l'opérateur produit de convolution :

$$\Delta(I * G) = (\Delta I) * G = I * (\Delta G) \tag{10}$$

Nous pourrons donc appliquer en une seule étape le filtre laplacien et le filtrage préalable G. En général, on choisit pour G un filtrage gaussien :

$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}} \Rightarrow \Delta G(x,y) = \frac{1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right) e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{11}$$

Le paramètre σ sert a régler la résolution à laquelle les contours sont détectés. Son choix est très difficile a priori car il est peu probable qu'une valeur unique permette de représenter efficacement toutes les résolutions présentes dans une image.

Dans le cas discret, la plus simple approximation du Laplacien d'une Gaussienne est le filtre suivant connu sous le nom de masque Laplacien :

$$\Delta G \to \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \tag{12}$$

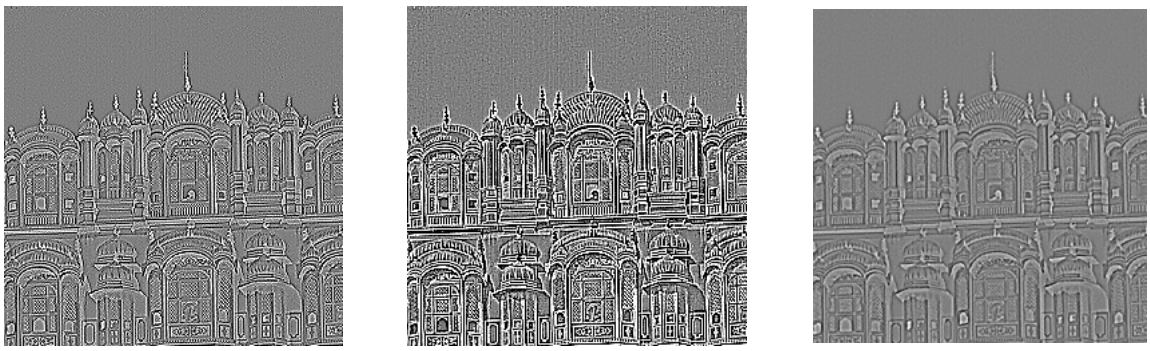


FIGURE 7 – Filtrage laplacien de base, avec masque laplacien 3x3 et avec noyau laplacien 5x5

2.2.3 Travail informatique

Dans un premier temps appliquer le filtre laplacien de base puis le masque laplacien aux diverses images.

Nous nous proposons ensuite de crée notre propre noyau de filtrage gaussien. Écrire une fonction *def* **noyaugau**(*sigma*) : qui prend en argument une valeur de σ donnée et retourne un noyau 5x5 tel que

$$NoyauG[i,j] = \frac{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(i-2)^2+(j-2)^2}{2\sigma^2}}}{\sum_{i,j} NoyauG[i,j]} \tag{13}$$

Utiliser ce filtre en pré-traitement du filtre laplacien de base (on pourra prendre $\sigma = 0.7$) et comparer avec le résultat donné par le masque laplacien.

Nous nous proposons ici d'étudier ici deux post-traitements :

- Le **seuillage** consiste à remplacer toutes des valeurs de la matrice traitée inférieures à *seuil* par 0. Cette opération a pour but d'éliminer le bruit généré par des dérivées secondes de trop petites variations de l'intensité de l'image. C'est un filtrage passe-haut.

  Écrire une fonction *def* **seuillage**(M, *seuil*) : prenant en entrée une matrice M déjà filtrée et un seuil de coupure *seuil* et retournant la matrice seuillée.
- Le **zero-crossing** consiste à détecter les changements de signe de la matrice filtrée (donc les changements de signe du laplacien). Pour cela, on crée une matrice I_p dont les éléments valent 1 si le laplacien de l'image est positif et 0 sinon. On crée ensuite une matrice I_z dont les éléments valent 1 si l'élément correspondant de I_p correspond à une transition $0 \to 1$ ou $1 \to 0$.

  Écrire une fonction *def* **zerocrossing**(M) : qui prend le laplacien de la matrice en entrée et retourne la matrice des zero-crossing

On peut bien sûr combiner les deux post-traitements.

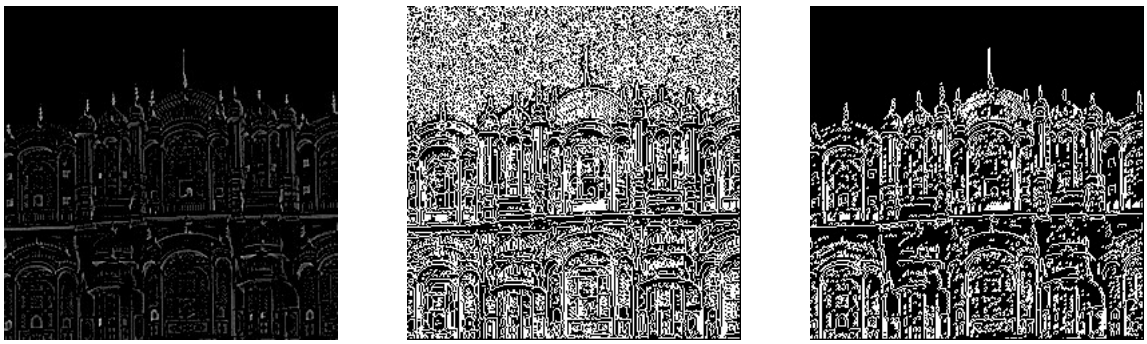


FIGURE 8 – Détection avec seuil=20, zero-crossing et combinaison des deux

Nous venons de voir un petit aperçu des premiers traitements de détection de contours. Il est évident que le travail ne s'arrête pas : il va falloir nettoyer les contours de manière plus fine, essayer d'éviter les interruptions de contour (par un seuillage par hystérésis et création d'un graphe d'adjacence entre les diverses composantes connexes d'un contour.

Ensuite vient tout le travail de reconnaissance de forme, de détection d'angles, de segmentation...

Le point principal à retenir est qu'il n'y a pas de méthode miracle qui fonctionne pour tous les contours de toutes les images. Il faut travailler par essai successifs en adaptant les divers paramètres des divers filtres.

3 Une méthode de contours actifs : le snake

Le snake est une courbe paramétrée, où s est généralement l'abscisse curviligne (longueur de la courbe) :

$$v(s) = [x(s), y(s)] \quad \text{où} \quad s \in [a, b] \tag{14}$$

L'énergie du snake s'exprime par :

$$E_{snake} = \int_a^b \left(E_{interne}(v(s)) + E_{image}(v(s)) + E_{externe}(v(s))\right) ds \tag{15}$$

L'énergie interne du snake s'exprime en notant $\alpha(s)$ est le facteur d'élasticité et $\beta(s)$ le facteur de rigidité du contour - permettant d'obtenir des courbes plus ou moins lisses- par :

$$E_{interne}(v(s)) = \alpha(s) \left(\frac{dv}{ds}\right)^2 + \beta(s) \left(\frac{d^2v}{ds^2}\right)^2 \tag{16}$$

L'énergie potentielle liée à l'image représente les éléments sur l'image vers lesquels on veut attirer le snake. Cette énergie est donnée par :

$$E_{image}(v(s)) = -\lambda(s)|\vec{\nabla} I(v(s))|^2 \tag{17}$$

où $\lambda(s)$ dépend de l'image I initiale et $\vec{\nabla}$ est l'opérateur gradient. On peut faire précéder le gradient d'un filtrage passe-bas de l'image permettant d'obtenir des contours moins bruités et d'augmenter leur zone d'influence.

L'énergie externe (ou de contraintes) est définie par l'utilisateur selon les spécificités du problème. On pourrait ainsi, par exemple, imposer une distance minimale ou maximale entre deux points consécutifs du contour actif.

En l'absence de contraintes extérieures, on montre que pour minimiser l'énergie du snake, il faut, en considérant les coefficients α, β et λ constants que

$$-\alpha \left(\frac{d^2v(s)}{ds^2}\right) + \beta \left(\frac{d^4v(s)}{ds^4}\right) = F(v(s)) \quad \text{où} \quad F(v(s)) = \lambda\vec{\nabla}|\vec{\nabla} I(v(s))|^2 \tag{18}$$

Les dérivées de l'équation sont ensuite approximées par des différences finies. On les met alors sous forme matricielle, nous donnant ainsi le schéma d'évolution suivant :

$$AV = F \quad \text{où} \quad A = \begin{pmatrix} -2\alpha+6\beta & -\alpha-4\beta & \beta & 0 & 0 & \dots & \dots \\ -\alpha-4\beta & -2\alpha+6\beta & -\alpha-4\beta & \beta & 0 & \dots & \dots \\ \beta & -\alpha-4\beta & -2\alpha+6\beta & -\alpha-4\beta & \beta & \dots & \dots \\ 0 & \beta & -\alpha-4\beta & -2\alpha+6\beta & -\alpha-4\beta & \dots & \dots \\ 0 & 0 & \beta & -\alpha-4\beta & -2\alpha+6\beta & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \tag{19}$$

Le snake va évoluer au cours du temps sous l'effet des forces dérivant des trois types d'énergie :$-AV$ représente la force dérivant de l'énergie interne, F la force attractive exercée par l'image. Matriciellement, nous pouvons donc écrire si τ est le pas du temps qui contrôle l'évolution du snake :

$$\frac{dV}{dt} = F_{interne} + F_{image} \quad \text{d'où} \quad \frac{V[t] - V[t-1]}{\tau} = -AV[t] + F(V[t-1]) \tag{20}$$

Soit

$$(Id + \tau A)V[t] = V[t-1] + \tau F(V[t-1]) \Leftrightarrow V[t] = (Id + \tau A)^{-1}(V[t-1] + \tau F(V[t-1])) \tag{21}$$

Lorsque $V[t]$ et $V[t-1]$ sont très proches, on considère que la convergence et réalisée et on peut arrêter le processus.

Ce processus présente des problèmes :

- liés au paramétrage : la définition de l'énergie dépend de la manière dont on paramètre le snake. De plus, le contour initial doit être suffisamment proche de l'objet pour pouvoir converger, sinon il risque de s'effondrer sur lui même.
- liés à la topologie : le snake ainsi défini sera incapable de détecter distinctement deux objets sur une image (au mieux, les contours des deux objets seront liés). L'objet à détecter doit également être convexe, les snakes ayant du mal à rentrer dans les concavités.
- liés aux calculs : le calcul de la dérivée d'ordre 4 qui apparaît dans l'équation d'évolution pose des problèmes de discrétisation et d'instabilités numériques.

L'algorithme GVF plus récent limite quelque peu les inconvénients du snake classique.