BIMA

Mini rapport TP5

Kim-Anh Laura Nguyen
Arij Riabi
M1 DAC
Promo 2018-2019

Enseignant : Dominique Bérériat

Exercice 1 - Comparaison de filtres discrets du 1er et du 2nd ordre

1. **Filtre de Sobel**

```
function [mod_seuille] = filtre_sobel(I, seuil)
    Gx = [1, 0, -1; 2, 0, -2; 1, 0, -1]; % filtre de Sobel en x
    Gy = [1, 2, 1; 0, 0, 0; -1, -2, -1]; % filtre de Sobel en y
    Ix = convolution(I, Gx); % produit de convolution entre I et Gx
    Iy = convolution(I, Gy); % produit de convolution entre I et Gy
    module = sqrt(Ix.^2 + Iy.^2); % calcul du module du gradient estime
    % produit un filtre detecteur de contours du premier ordre
    mod_seuille = seuillerImage(module, seuil);
end
```

FIGURE 1 – Fonction `filtre_sobel`

2. **Filtre laplacien**

```
\centering
function [M] = filtre_laplacien(I,seuil)
    [n,m] = size(I);
    L = [0, 1, 0; 1, -4, 1; 0, 1, 0]; % masque laplacien
    IL = convolution(I, L); % produit de convolution entre I et L

    % matrice n x m dont chaque pixel p vaut 255 s'il correspond a un contour
    % 0 autrement
    M = zeros(n,m);

    % determination des passages par 0 de IL
    for i=2:n-1
        for j=2:m-1
            sous_mat = IL(i-1:i+1, j-1:j+1);
            max_IL = max(max(sous_mat));
            min_IL = min(min(sous_mat));

            if ((max_IL > 0) && (min_IL < 0) && ((max_IL -min_IL) > seuil))
                M(i,j) = 1;
            end
        end
    end
end
```

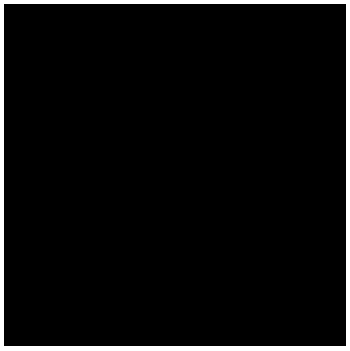
FIGURE 2 – Fonction `filtre_laplacien`

3. **Comparaison des deux filtres**

La figure 3 contient les images obtenues avec chaque filtre détecteur de contours sur l'image `lena.gif` (figure 3a).

On remarque que le filtre de Sobel (figure 3b) détecte les contours fins. En revanche, le masque laplacien (figure 3c) génère des contours épais et est sensible au bruit (les hautes fréquences correspondant à l'arrière-plan). Cela est dû à la double dérivation, qui amplifie davantage le bruit, ainsi qu'à l'erreur de quantification, liée au fait que l'on traite une fonction entière et discrète.

(a) Image originale

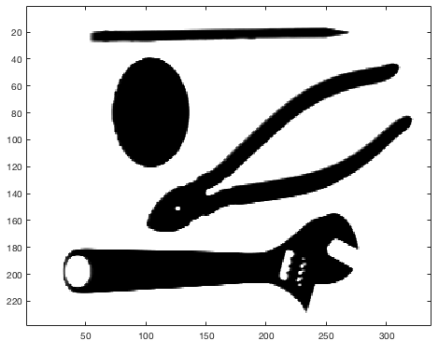


(b) Filtre détecteur de contours du premier ordre (seuil à 70)

(c) Filtre détecteur de contours du second ordre (seuil à 70)

FIGURE 3 – Application des filtres détecteur de contours du premier et du second ordre sur l'image `lena.gif`

Exercice 2 - Suppression de non maxima



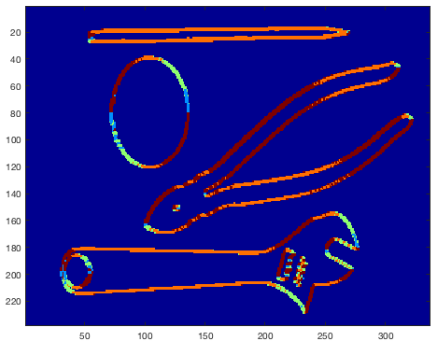


FIGURE 4 – Image `tools.gif` et image de l'orientation des gradients associés

```
function [new_Ig] = nms(Ig,Ior)
    % Ig : image du module du gradient
    % Ior : image de l'orientation des gradients
    % renvoie une image du module du gradient ou seuls
    % les extrema locaux sont conserves

    [n,m] = size(Ig);
    zeros_c = zeros(n,1); % colonne de zeros
    zeros_l = zeros(1, m+2); % ligne de zeros

    % Ig2 : matrice Ig dans laquelle on rajoute
    % de part et d'autre 2 lignes et 2 colonnes de zeros
    Ig2 = [zeros_c Ig zeros_c];
    Ig2 = [zeros_l; Ig2; zeros_l];
    new_Ig = zeros(n,m);

    for i=2:n+1
        for j=2:m+1
            or = Ior(i-1,j-1);
            v = Ig2(i,j);
            if or == 1 % ouest, est
                v1 = Ig2(i,j-1);
                v2 = Ig2(i,j+1);
            elseif or == 2 % nord-est, sud-ouest
                v1 = Ig2(i-1,j+1);
                v2 = Ig2(i+1,j-1);
            elseif or == 3 % nord, sud
                v1 = Ig2(i-1,j);
                v2 = Ig2(i+1,j);
            elseif or == 4 % nord-ouest, sud-est
                v1 = Ig2(i-1,j-1);
                v2 = Ig2(i+1,j+1);
            else
                continue
            end

            if (v > v1) && (v > v2)
                new_Ig(i-1,j-1) = 255;
            end
        end
    end
end
```

FIGURE 5 – Fonction `nms`

Comme l'on met à zéro la norme du gradient pour les pixels non maxima locaux, on obtient donc des contours d'épaisseur 1 pixel.

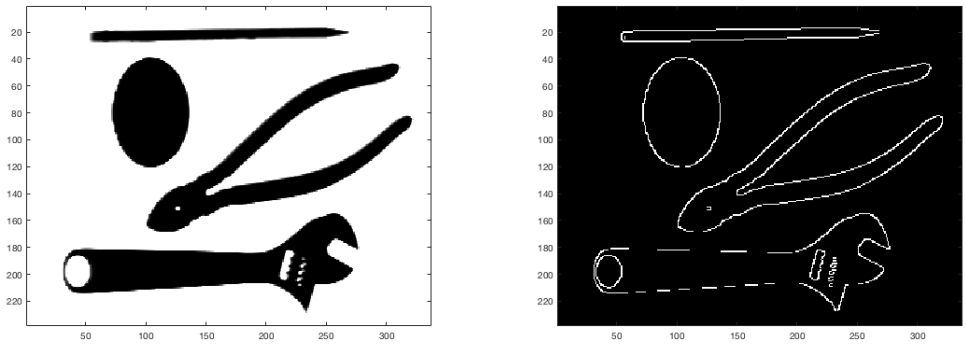


FIGURE 6 – Image `tools.gif` et image du module du gradient avec suppression de non maxima, sans lissage gaussien

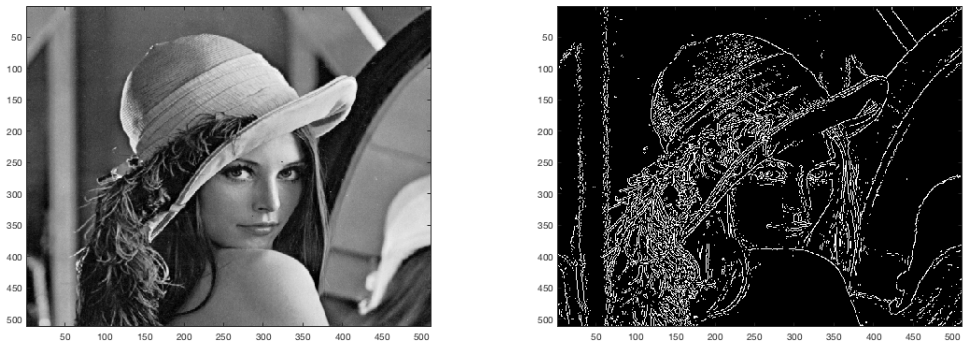


FIGURE 7 – Image `lena.gif` et image du module du gradient avec suppression de non maxima, sans lissage gaussien

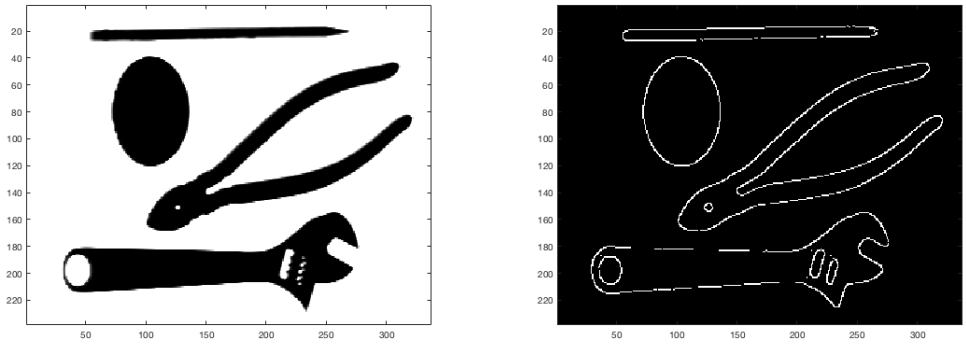


FIGURE 8 – Image `tools.gif` et image du module du gradient avec suppression de non maxima, et lissage gaussien ($\sigma = 3$)

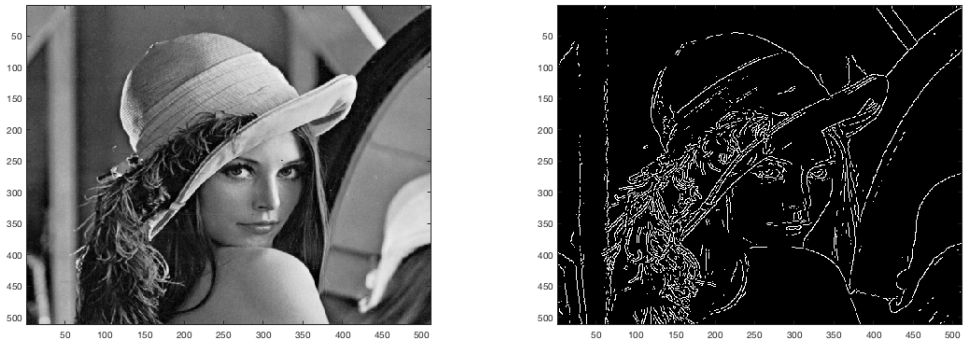
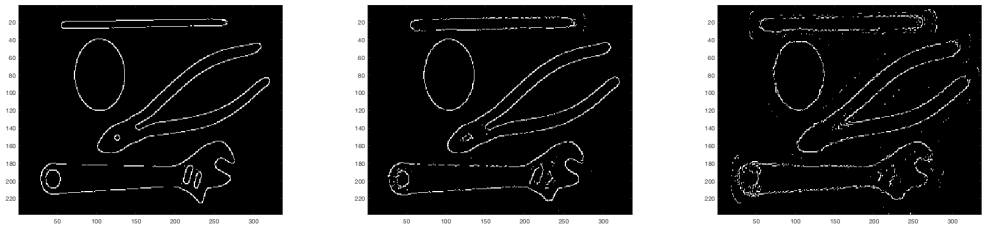


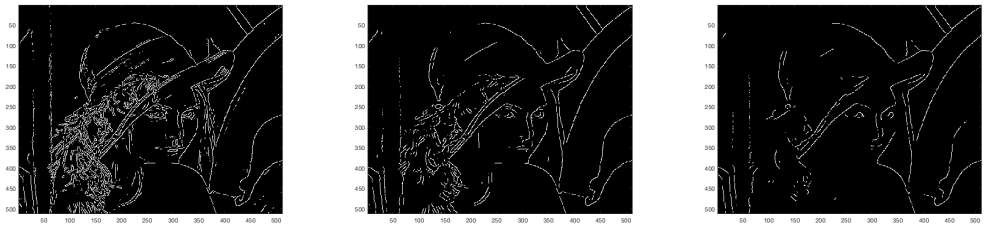
FIGURE 9 – Image `lena.gif` et image du module du gradient avec suppression de non maxima, et lissage gaussien ($\sigma = 1$)

On remarque sur les figures 8 et 9 qu'un lissage gaussien (à $\sigma = 3$ pour `tools.gif` et $\sigma = 1$ pour `lena.gif`) appliqué avant la différentiation permet de réduire le bruit : les variations locales sont filtrées et les contours dominants restent.



(a) $\sigma = 3$ (b) $\sigma = 5$ (c) $\sigma = 7$

FIGURE 10 – Images du module du gradient de `tools.gif` avec suppression de non maxima, et lissage gaussien



(a) $\sigma = 1$ (b) $\sigma = 2$ (c) $\sigma = 3$

FIGURE 11 – Images du module du gradient de `lena.gif` avec suppression de non maxima, et lissage gaussien

Exercice 3 - Influence du lissage dans la détection de contours

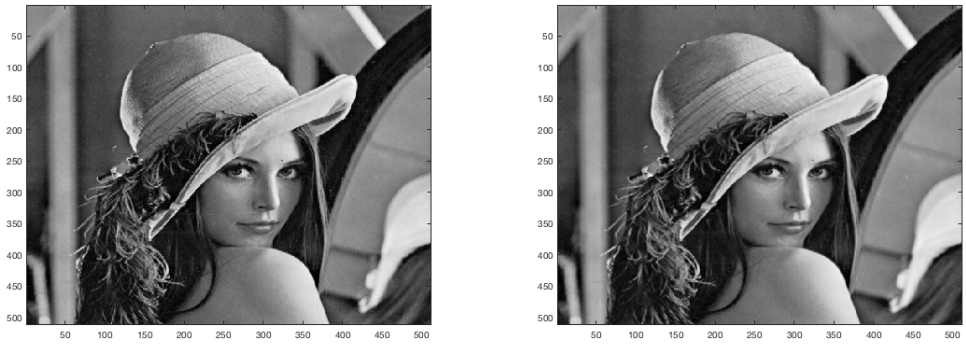


FIGURE 12 – Image `lena.gif` originale et lissage gaussien avec $\sigma = 0.5$

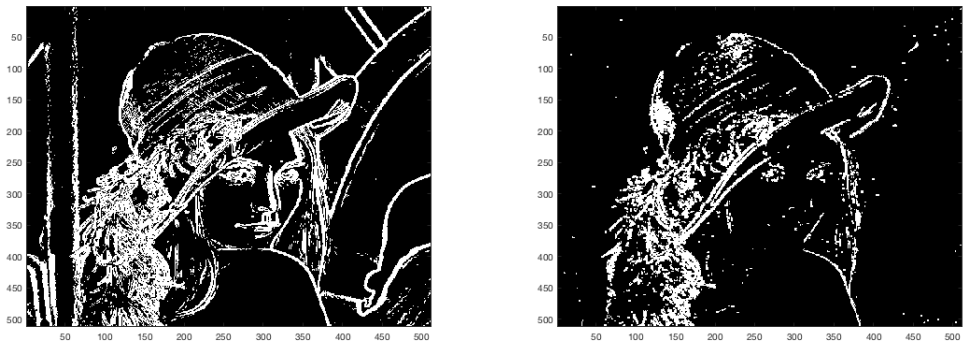


FIGURE 13 – Application des filtres de Sobel et laplacien à l'image `lena.gif` lissée ($\sigma = 0.5$)

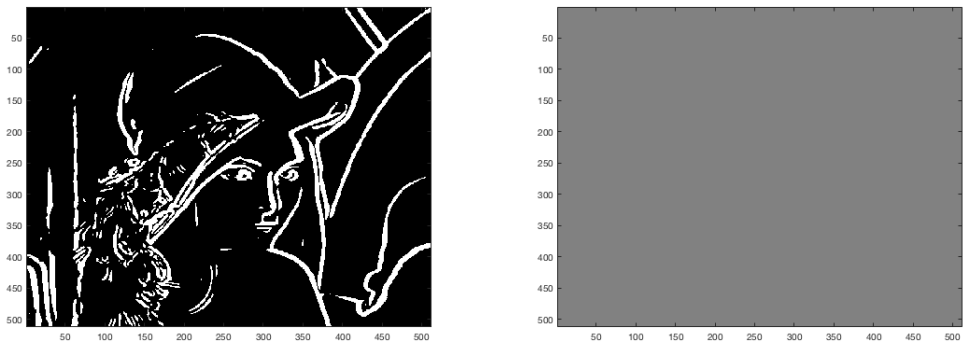


FIGURE 14 – Application des filtres de Sobel et laplacien à l'image `lena.gif` lissée ($\sigma = 2$)

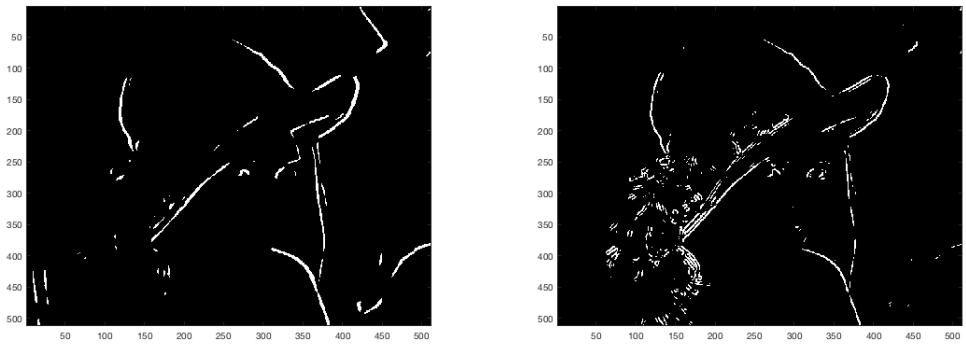


FIGURE 15 – Application des filtres de Sobel et laplacien à l'image `lena.gif` lissée ($\sigma = 2$) avec des seuils respectifs de 150 et 10