Traitement du signal et des images : Projet mini boîte a outils pour traitement d'image

Chaolei Cai
Université Paris Vincennes St-Denis
UFR mathématiques, informatique,
technologies sciences de l'information
L3 Informatique

April 8, 2020

Contents

1 Présentation

Ce document est mon rapport pour le cours de Traitement du signal et des images enseigné par M.Boubchir au 3ème année de la Licence Informatique. Le but du projet est donc crée un petit programme avec une interface graphique, permettant ainsi de visualiser rapidement certain traitement applicable sur l'image.

2 Dépendances

Le projet a été écrite en Scilab 6.0.2, il nécessite la librairie IPCV (Image Processing and Computer Vision Toolbox).
L'interface graphique a été écrite via l'outils guibuilder, si vous rencontrez des problèmes d'exécution, vérifier la version de Scilab que vous possèdez et vérifier la présence de ces 2 librairies. Recemment, il y a eu une mise à jour de Scilab en 6.1.0, je n'ai pas encore tester le programme sous cette version car le developpement a été faite en 6.0.2

La compatibilité avec une version Scilab 5.X.X est sans doute possible, mais il faudra charger le module SIVP (Scilab Image and Processing Video).
Il faut patienter un brief moment après le click sur un boutton car les calcules sont longues.

3 Apperçue génerale de l'interface

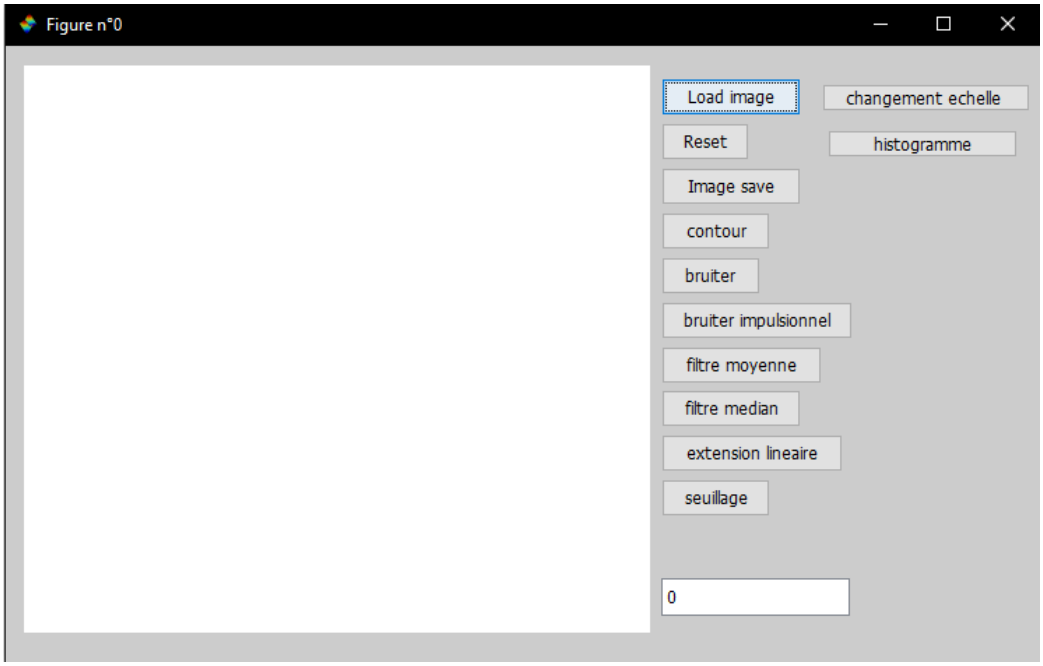


Figure 1: Page d'accueil

Voici donc la page d'accueil de mon programme, pour ouvrir cette interface, ouvrez Scilab, depuis le navigateur de fichier, il faut aller vers le repertoire où se trouve "gui.sce" et "traitement.sce".
Depuis la console de scilab, tapez la commande suivante "exec gui.sce".

4 Présentations des fonctionalités

4.1 Load image, Reset, Image save

Comme leurs noms l'indiquent, ces 3 bouttons permettent de charger une image, recharger l'image et de sauvergarder l'image. Dès lors, une interface de navigateur de fichier s'affiche, cette fonctionalités ont été réalisé via les fonctions "uigetfiles" et "uiputfiles"

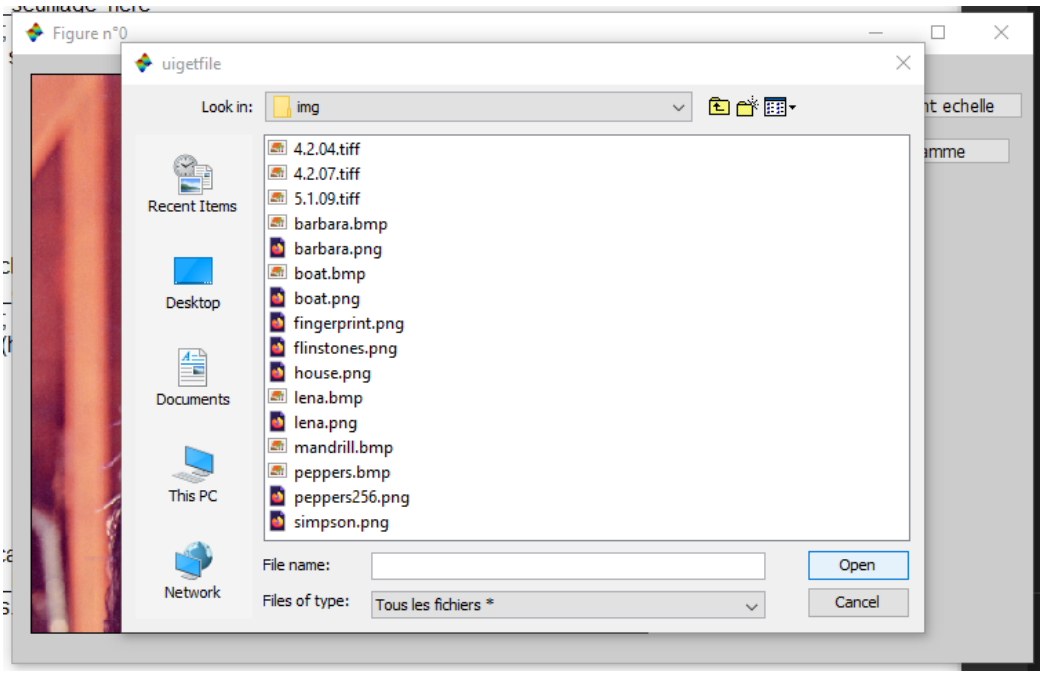


Figure 2: Interface de chargement & sauvergarde

4.2 Contour

Le boutton contour permet de d'afficher le contour des formes contenue dans l'image, via la norme du gradient discret, c'est une manière possible d'implémenter le filtre de Sobel enseigné en cours.

4.2.1 Norme du gradient

La fonction suivante renvoie la norme du gradient discret en fonction d'une image, il faut alors préciser sa position i, j. Les dimensions sizeX, sizeY et sizeZ de l'image, ainsi qu'une constance à appliquer pour pondérer le résultat.

```
1      function [v] = norme_gradient(img,i,j, sizeX, sizeY, sizeZ,cst)
2      if i < sizeX then
3          a = img(i+1, j, sizeZ);
4      else
5          a = 0;
6      end
7
8      if i > 1 then
9          b = img(i-1, j, sizeZ);
10     else
11         b = 0;
12     end
13
14     if j < sizeY then
15         c = img(i, j+1, sizeZ);
16     else
17         c = 0;
18     end
19
20     if j > 1 then
21         d = img(i, j-1, sizeZ);
22     else
23         d = 0;
24     end
25     v = cst * sqrt( (a-b)**2 + (c-d)**2);
26 endfunction
```

```
1  function [M] = im_contour(img, cst)
2  [sizeX, sizeY, sizeZ] = size(img);
3  M = zeros(sizeY, sizeX,sizeZ);
4  for k = 1 :sizeZ
5      for i = 1 :sizeY
6          for j = 1:sizeX
```

```
7               //disp(i,j);
8               M(i,j, k) = norme_gradient(img, i, j, sizeX, sizeY, k, cst);
9           end
10      end
11  end
12  endfunction
```

Enfin la fonction im_contour renvoie l'image du contour à partir d'une image et du coefficient à appliquer.

La fonction ne fait pas de distinction quelque soit la nature de l'image, il peut être en noire et blanc ou en couleur, cenpendant, le traitement des images en couleur est plus lente car nécessite plus de calcules.

4.3 Bruitage

Pour le bruitage, 2 options sont disponible, la première est le bruitage constant, il faut saisir une valeur dans le champs de saisit en bas, puis cliquer sur le boutton "bruiter", la fonction ajoute un bruit géneré aléatoirement et pondérer par la valeur saisit.
Le bruit impulsionnel est plus sophistiqué, ce n'est plus une valeur comme paramètre de bruit mais un pourcentage, la fonction applique un bruit aléatoire sur cette pourcentage d'image aléatoirement, par exemple si la valeur saisit est de 50, alors 50 pourcentage de l'image recevront un bruit aléatoire.

```
1       function [Ub] = bruite(U, s)
2       // Description of bruite(U, s)
3           Ub = rand(size(U,1), size(U,2),size(U,3), 'normal');
4           Ub = Ub * s + U;
5       endfunction
6
7
8       function [Uimp] = bruite_imp(U, p)
9       // Description of bruite(U, s)
10      I = rand(U);
11      //disp(I);
12      Uimp = 255*rand(U).*(I <p/100) + (I>=p/100).*U;
13      endfunction
```

4.4 Débruitage via la méthode de filtre

Le débruitage est possible via 2 filtre: le filtre moyenne ou le filtre médian, il faut faire attention au champs de saisit, car le débruitage nécessite la dimensions du filtre, par exemple la valeur 2 donneras un filtre de 2 * f +1 soit 5x5 en dimensions. Plus le filtre est grand, plus la fonction est longue en pour ses calculs.

```
function [M] = im_moyenne(U,f)
// Description of im_moyenne(U, f)
nb = (2*f +1)^2;
[sizeY, sizeX, sizeZ] = size(U);
M = zeros(U);
for k = 1:sizeZ
for i = 1:sizeY
    for j = 1:sizeX
        M(i,j, k) = sum(im_extract(U, i, j , f, k))/nb;
    end
end
end
endfunction

function [M] = im_median(U,f)
    // Description of im_median(U, f)
    nb = (2*f +1)^2;
    [sizeY, sizeX, sizeZ] = size(U);
    M = zeros(U);

    for k = 1:sizeZ
    for i = 1:sizeY
        for j = 1:sizeX
            M(i,j,k) = median(im_extract(U, i, j , f, k));
        end
    end
end
endfunction
```

sum et médian sont des fonctions de base de Scilab, im_extract est une fonction que j'ai écrite pour extraire une sous matrice depuis ses cordonnées et

ses dimensions.

4.5 Extension linéaire

Extension linéaire est une technique qui permet d'améliorer le contraste d'une image, il permet notamment d'étirer l'histogramme d'une image sur ses valeurs limites non utilisé.

```
function [I] = extension_lineaire(U)
// Description of extension_lineaire(U)
LUT = zeros(256, size(U,3));
for k = 1 : size(U,3)
for ng = 1:256
    LUT(ng, k) = 255 * (ng - min(U(:,:,k))) / (max(U(:,:,k)) - min(U(:,:,k)));
    //mprintf("ng = %d , lut(ng) = %d\n", ng, 255 * (ng - min(U(:,:,k))) / (max
end
end
I = zeros(U);
for k = 1: size(U,3);
for i = 1: size(U,1)
    for j = 1:size(U,2)
        I(i,j,k) = LUT(U(i,j,k), k);
    end
end
end

endfunction
```

L'extension linéaire se fait par une look up table, on créer d'abord cette LUT via les valeurs minimal et maximal, puis une image resultant est géneré via la consultation de cette LUT.

4.6 Seuillage

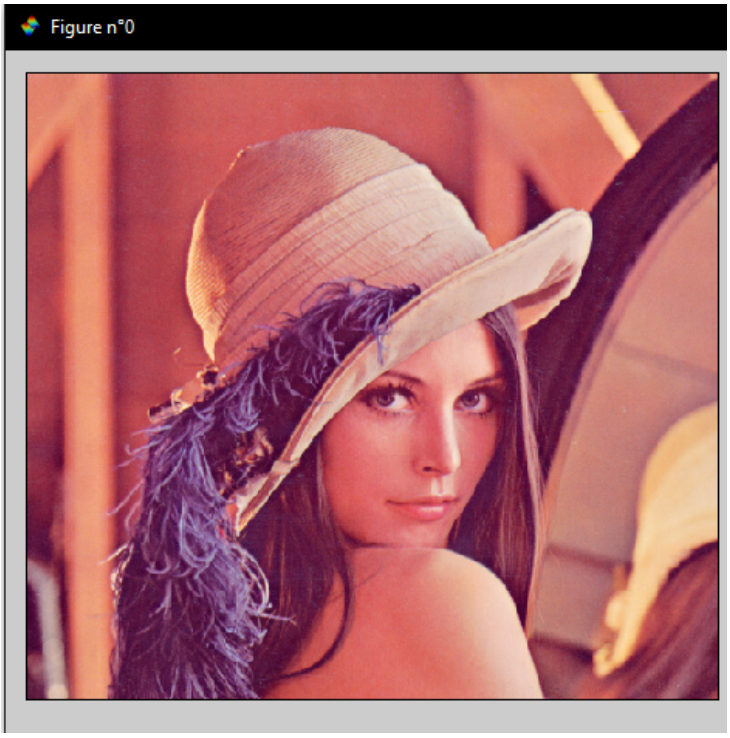
Le Seuillage est une technique qui étire l'image vers les valeurs limites selon une valeur de réference, on obtiens alors une image noire et blanc, cette fonctionalités ne marche que pour pour les images en niveau de gris.

4.7 changement d'echelle

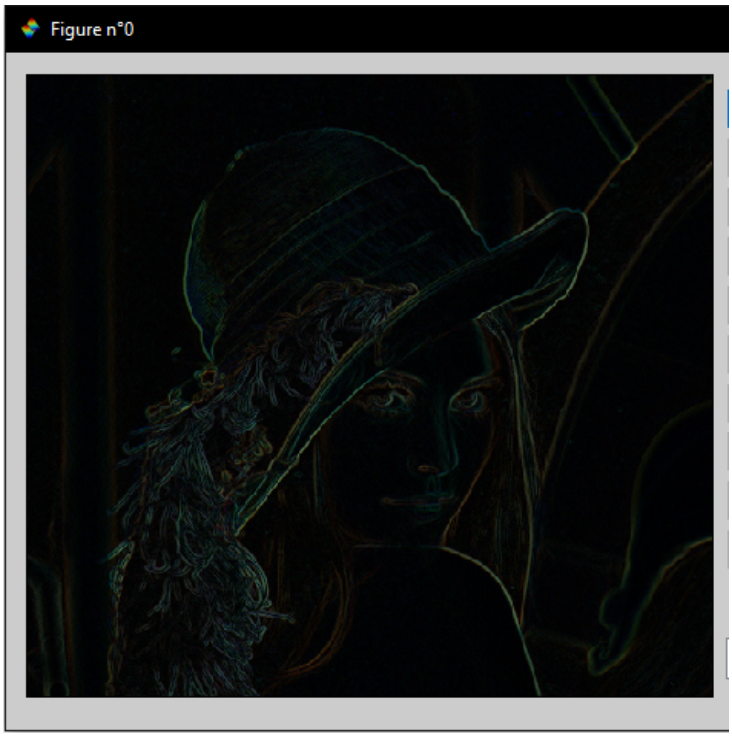
Cette fonctionalités permet de change l'échelle de l'image, cenpendant il ne donne pas de résultat sur l'interface graphique, essayer de sauvergarder l'image et de comparer ainsi les résultats.

4.8 histogramme

Cette fonctionalités permet d'afficher l'histogramme d'une image quelque soit sa nature.



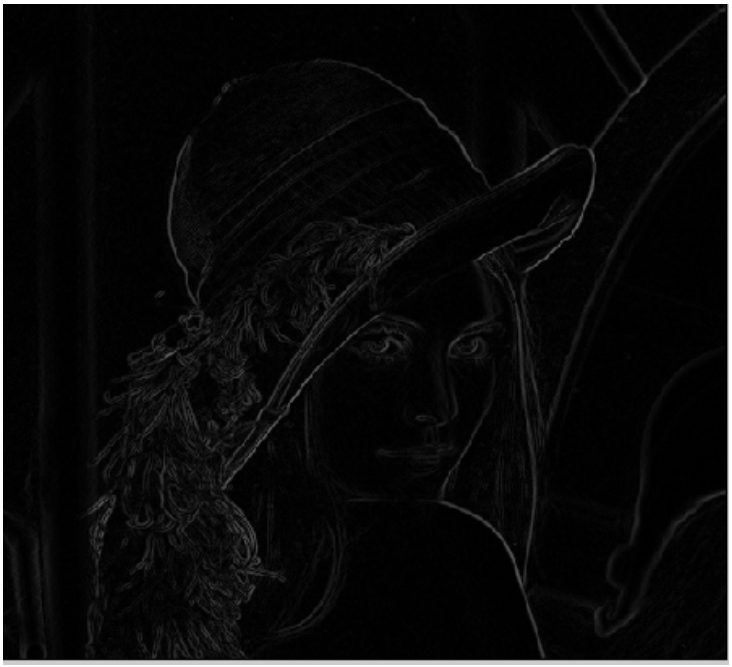
(a) Image initial: 4.2.04.tiff



(b) L'image du contour après traitement

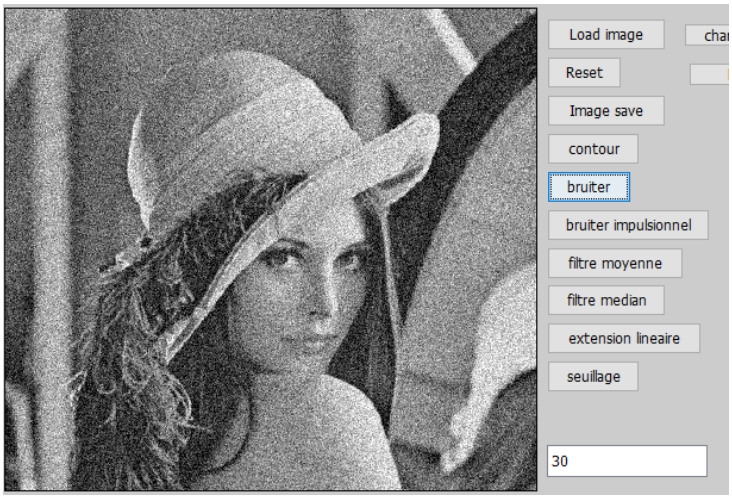
Figure 3: Contour sur une image couleur

(a) Image initial: lena.png

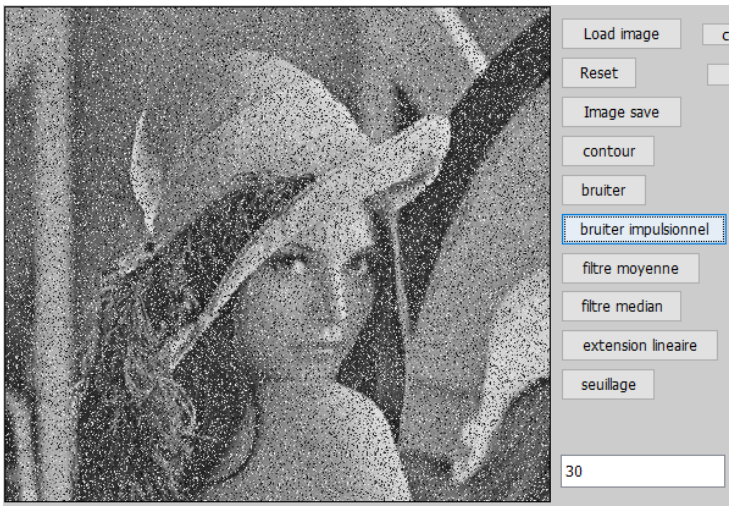


(b) L'image du contour après traitement

Figure 4: Contour sur une image noire et blanc



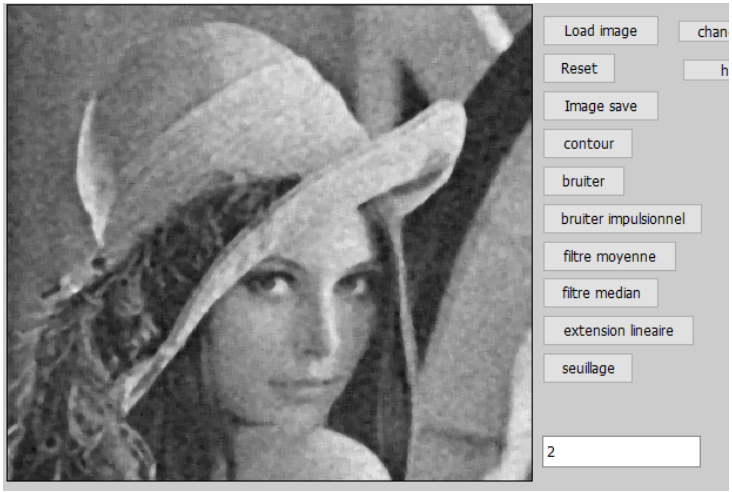
(a) Bruitage basique



(b) Bruitage impulsionnel

Figure 5: Bruitages

(a) Filtrage moyenne sur un filtre de 5x5



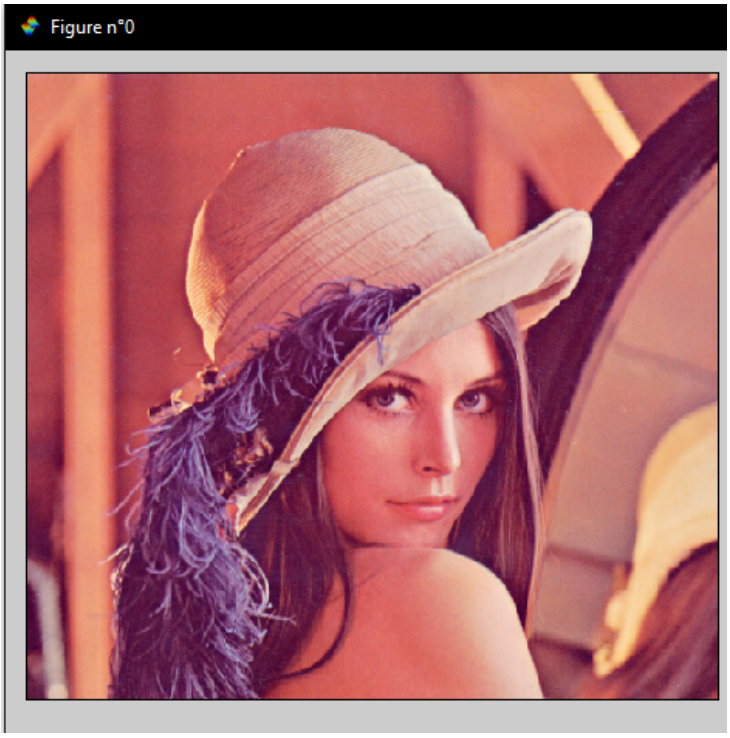
(b) Filtrage median sur un filtre de 5x5

Figure 6: Application des 2 filtres sur une image bruité

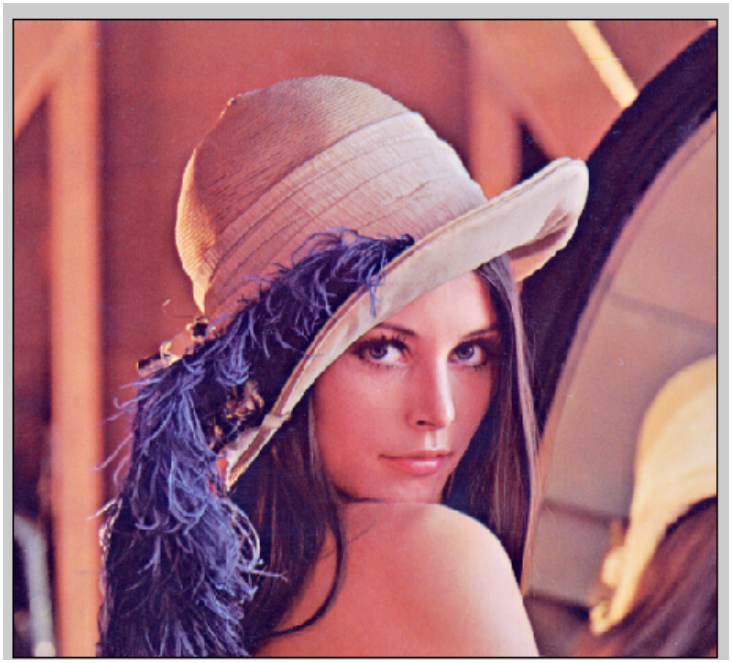
(a) Image initial

(b) Image après filtrage

Figure 7: Extension linéaire sur une image noire et blanc



(a) Image initial



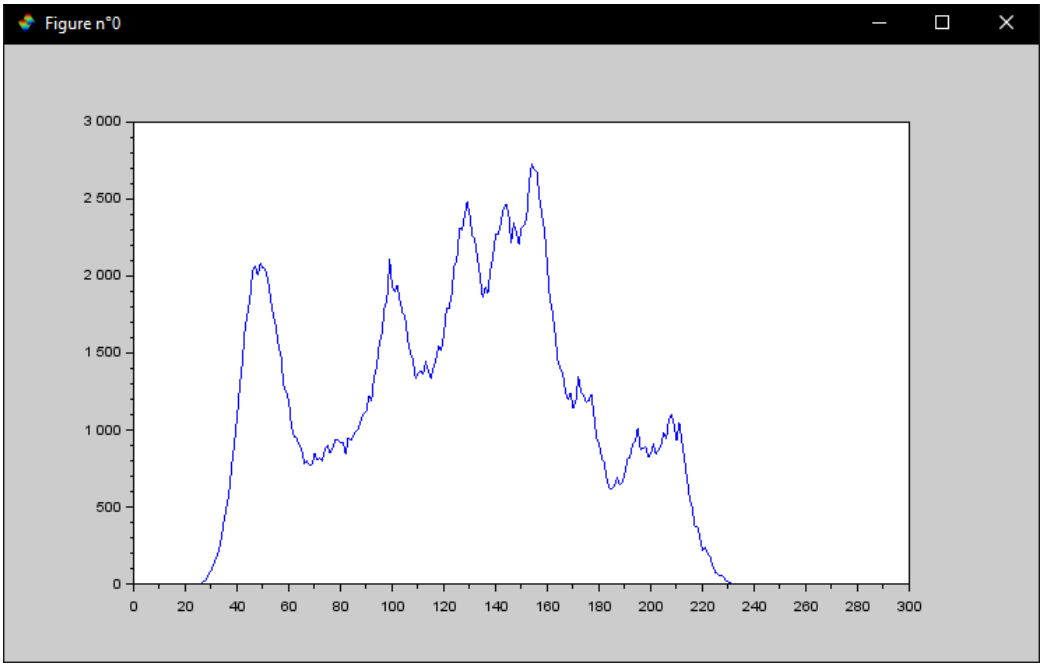
(b) Image après filtrage

Figure 8: Extension linéaire sur une image couleur

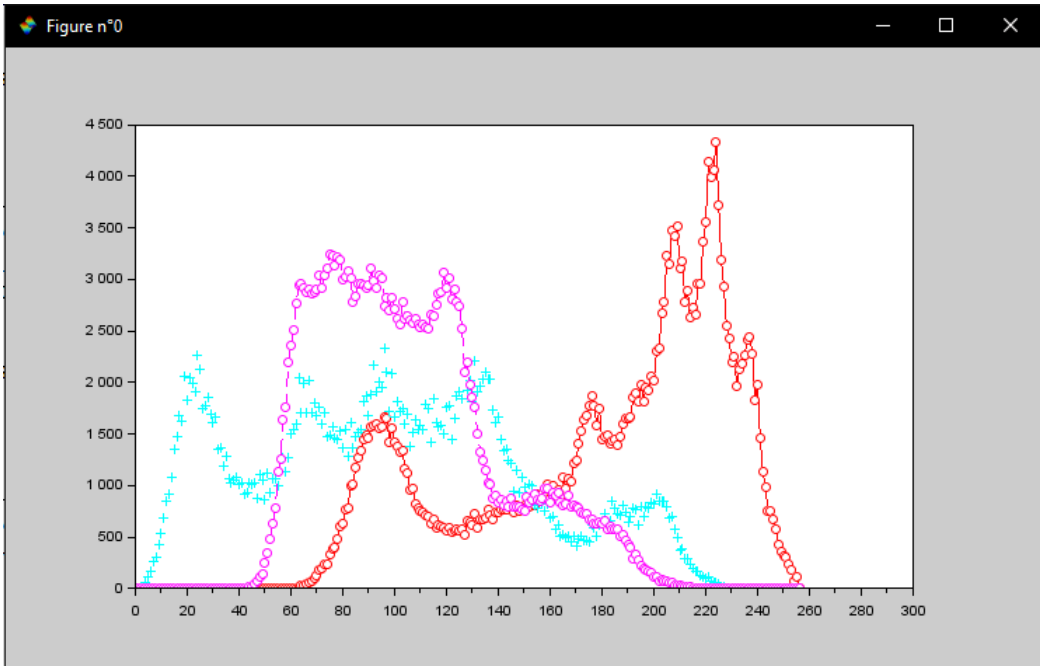
(a) Image initial

(b) Image après seuillage

Figure 9: Résultat du seuillage



(a) histogramme noire et blanc



(b) histogramme couleur

Figure 10: Résultat de histogramme