Antoine FONTANEZ

Rapport Matlab

TD 1

1) Fonction Binarize

```
function y=binarize(m, s)
[r, c] = size(m);
y = m;
for i=1:r
   for j=1:c
      if m(i, j) < s
         y(i, j) = 0;
      else
         y(i, j) = 255;
      end
   end
end
```

2) Fonction Paterne

```
function y=pat(valeur_fond, valeur_pattern, hauteur, largeur, xh, yh, xb, yb)
y = zeros(hauteur, largeur);
for i=1:hauteur
   for j=1:largeur
      if i>=yh && i<=yb && j>=xh && j<=xb
         y(i, j) = valeur_pattern;
      else
         y(i, j) = valeur_fond;
      end
   end
end
```

3) Fonctions addition soustraction et inverse

```
function y=addition(m1, m2)
[r, c] = size(m1);
y = m1;
for i=1:r
   for j=1:c
      y(i, j)=min(m1(i,j)+m2(i,j), 255);
   end
end

function y=substraction(m1, m2)
[r, c] = size(m1);
y = m1;
for i=1:r
   for j=1:c
      y(i, j)=max(m1(i,j)-m2(i,j), 0);
   end
end
```

```
function y=reverse(m)
[r, c] = size(m);
y = m;
for i=1:r
    for j=1:c
        y(i,j)=255-y(i,j);
    end
end
```

Exemple avec la fonction paterne :

J=pat(0, 255, 30, 50, 10, 7, 15, 9)

4) Tests sur images

On applique binarize sur Officier.png avec différents seuils :

50

100

150

200

Addition et soustraction d’Officer et Street_Malte

Inversions d’Officier et Street_Malte

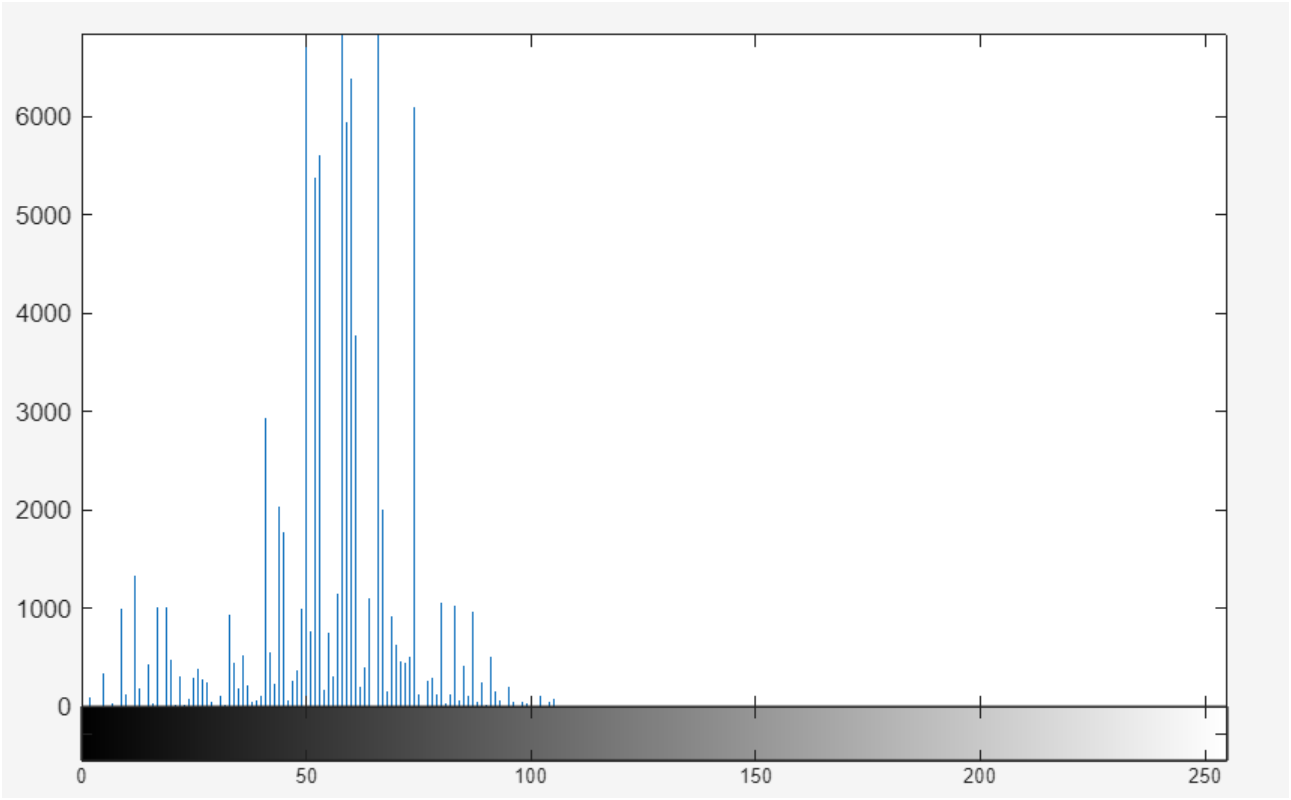
TD 2

1) L’image I est une image de voiture en .ppm.

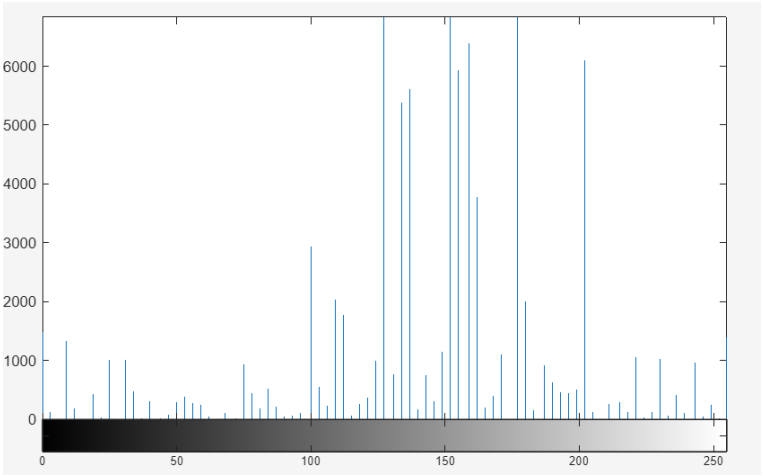
Ing=im2gray(I)

JPEG, PNG et TIFF sont des formats d’image supportés par Matlab

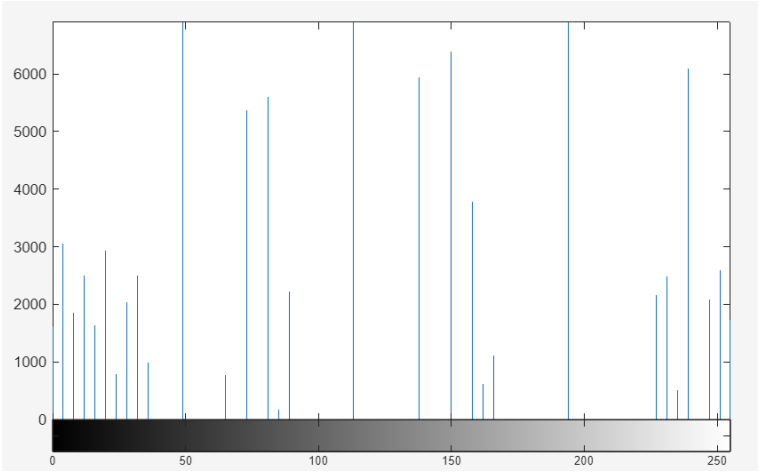
2)



L’histogramme est principalement composé de valeurs sombres, inférieures à 100.



Le imadjust augmente le contraste de l’image en « étirant » les niveaux de gris. Les niveaux de gris vont maintenant de 0 à 255.



Le histeq étire également l’histogramme mais en plus de ça il essaye d’égaliser les valeurs de gris, que les barres aient la même taille. Cela cause un contraste encore plus fort que le imadjust.

3) Images J et K :

Images K1 et L :

En remplaçant 0.02 par 0.08, le taux de bruit est bien plus élevé. Il y a plus de « sel et poivre ».

L'image K est floutée et bruitée alors que l'image J est relativement nette, comme l'image de base. imfilter(fspecial()) applique un filtre sur l'image, fspecial() correspond à un filtre prédéfini, dans notre cas il s'agit d'un filtre moyenneur 'average'.

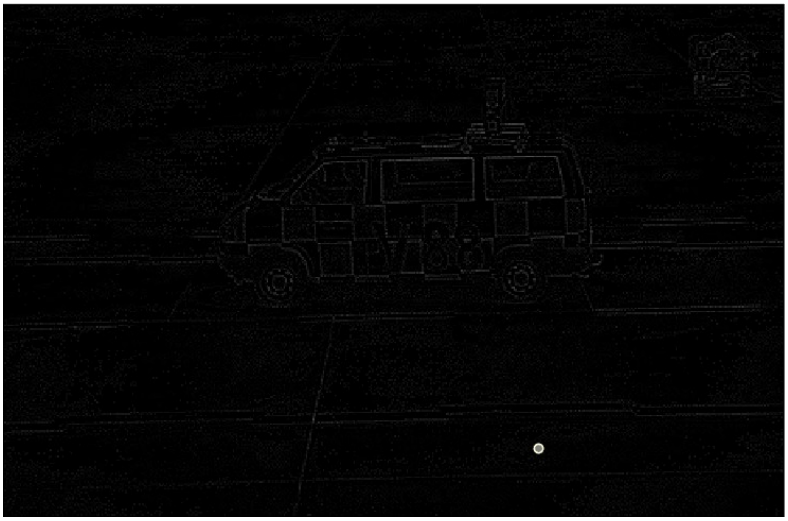
Medfilt2 applique un filtre médian 3x3 sur chaque pixel de l'image, ce qui élimine le bruit et lisse l'image. C'est la raison pour laquelle l'image L ressemble beaucoup à l'image de base non bruitée.

4) Jg=imnoise(I, 'gaussian', 0, 0.01)
Jg2=imfilter(Jg, fspecial('gaussian', sigma=1)
Jg3=imfilter(Jg, fspecial('gaussian', sigma=3)
figure, imshow(Jg)
figure, imshow(Jg2)
figure, imshow(Jg3)

Le bruit gaussien ajoute de petites particules sur l’image de base. Le filtre gaussien sigma 1 floute l’image, à l’inverse le gaussien sigma 3 rend l’image plus nette.

5) Application d'un filtre laplacian :

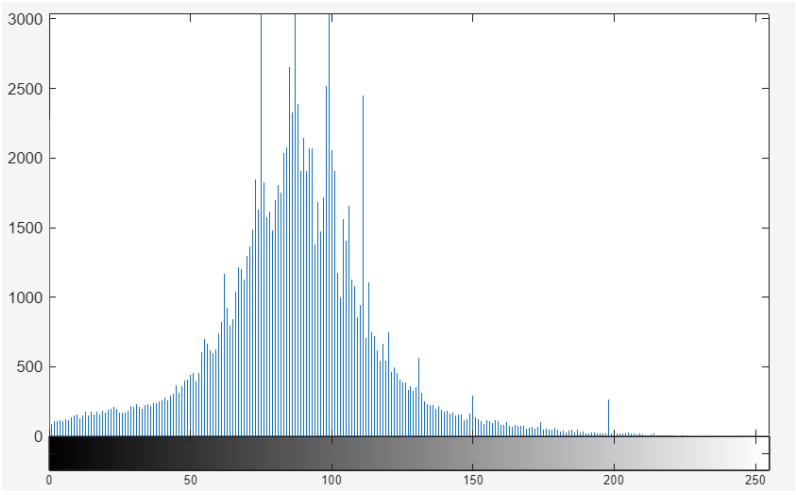
imshow(imfilter(Ing, fspecial("laplacian", 0.2)))



Le filtre semble délimiter les contours de l'image.

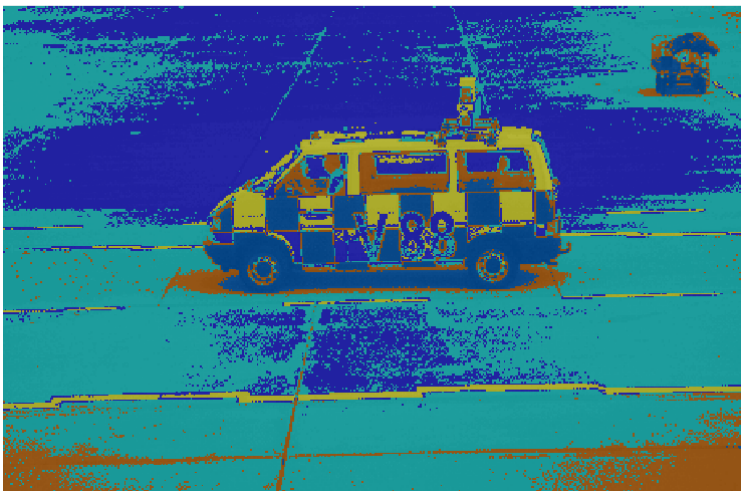
6) On filtre l'image avec un masque

```
h=[-0.25, -1, -0.25; -1, 6.5, -1; -0.25, -1, -0.25]
filtered = imfilter(Ing, h)
figure, imhist(filtered)
figure, imshow(filtered)
```



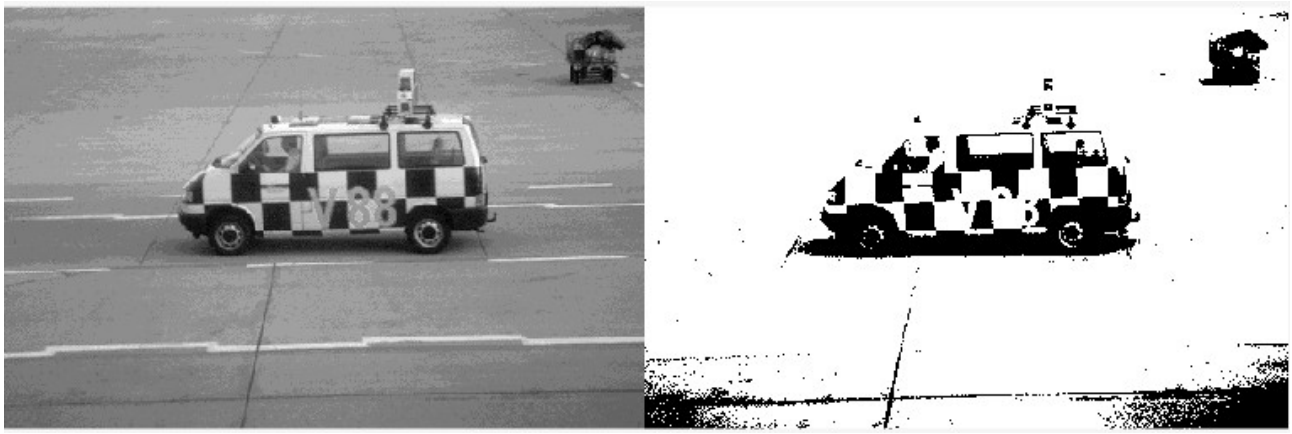
Les niveaux de gris sont étalés. Le contraste est amélioré : l'histogramme occupe une plage plus large de niveaux de gris. Les détails deviennent plus visibles dans les zones autrefois plates.

8) B et B1 :



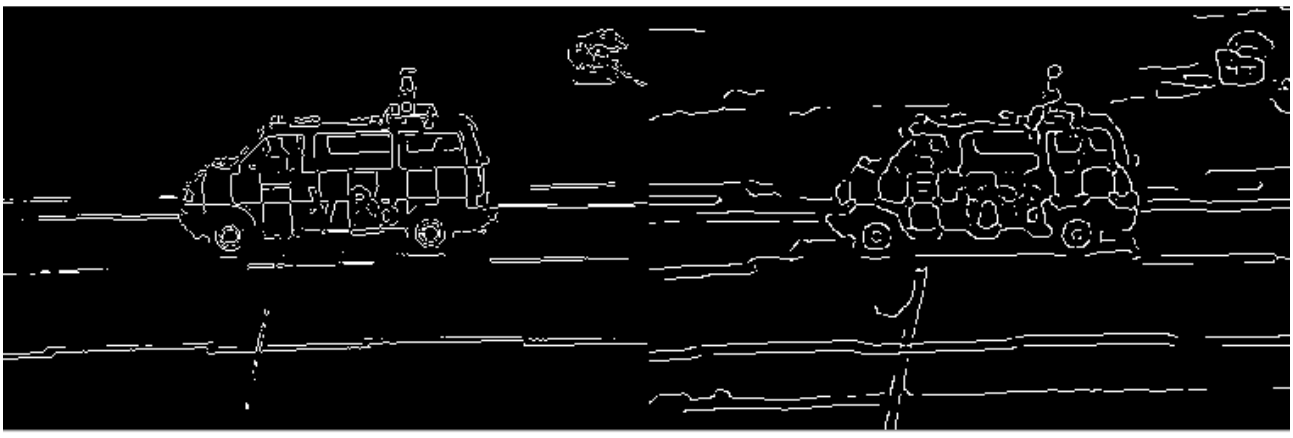
On remarque que B possède plus de couleurs que B1, ce qui est normale car sur la première image nous avons appliqué un Kmeans à 5 zones, tandis que la deuxième n'en a que 3.

9) level est égal à 0.1765



La méthode fondamentale entre Otsu et Kmeans est que Otsu fait toujours 2 catégories, en utilisant un seuil (level) calculé de façon à maximiser la variance entre ces deux catégories.

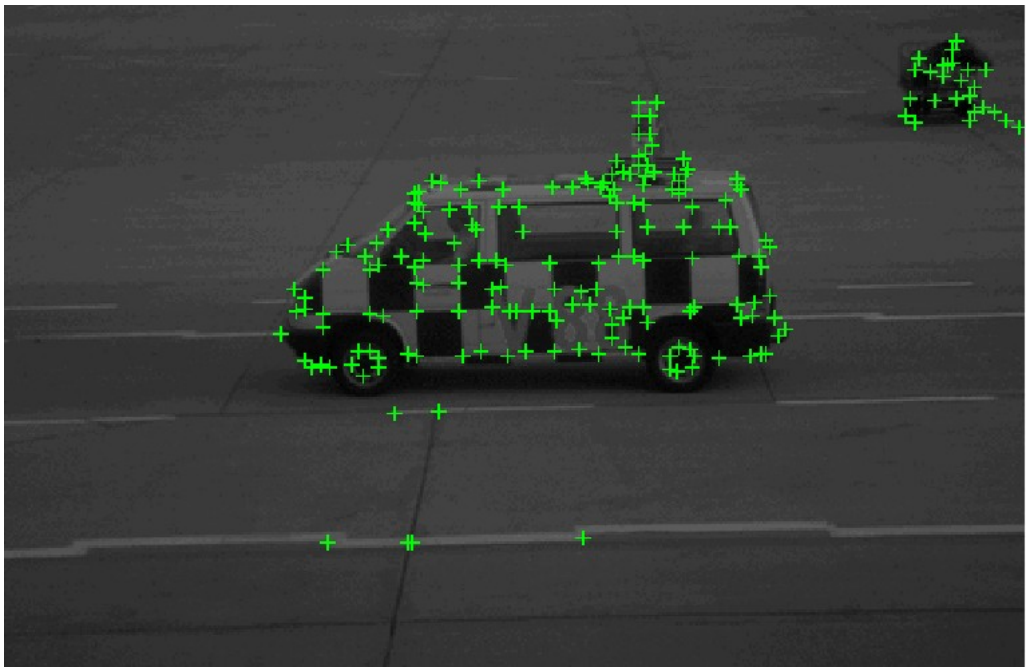
10) B1=edge(Ing, 'sobel')
B2=edge(Ing, 'canny', 0.1, 3)
imshowpair(B1, B2, 'montage')



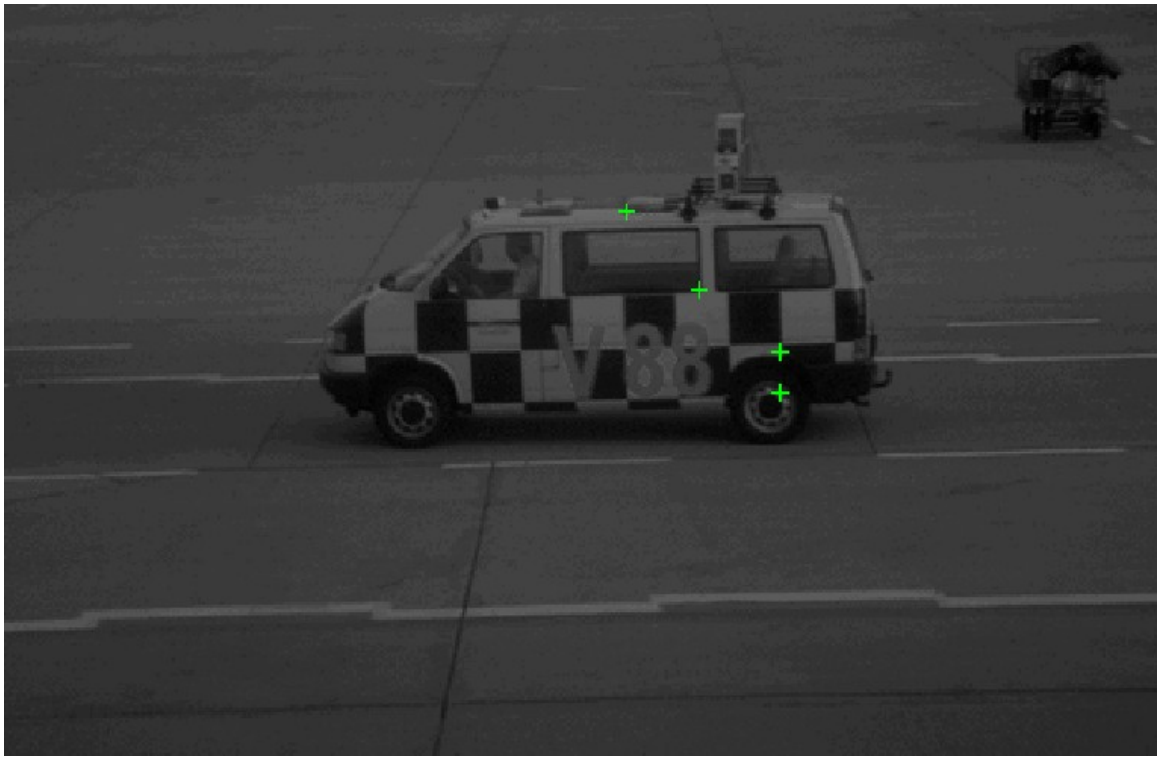
Sobel utilise du bruit, Canny est plus précis mais doit être paramétré.

11)

12) corners = corner(Ing)
imshow(insertMarker(Ing, corners))



harrisCorners = detectFASTFeatures(Ing)
imshow(insertMarker(Ing, harrisCorners.selectStrongest(200)))



On constate que la détection rapide détecte très peu de points.

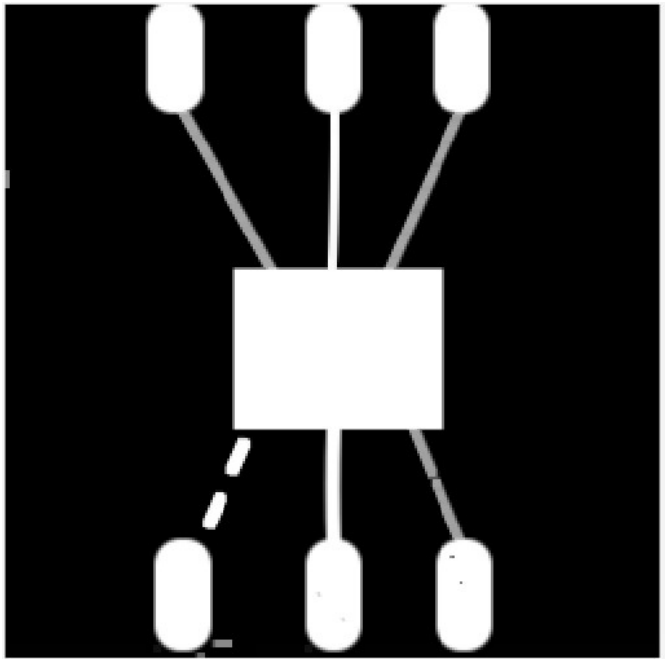
A l’inverse, Corners détecte beaucoup de faux points, on peut améliorer ça en mettant une limite de points à afficher en paramètre de la fonction corners. Par exemple corners(Ing, 20) ne prendra que les 20 points les plus importants

13) [cropped,rect] = imcrop(Ing);

```
corr = normxcorr2(cropped, Ing)
maxCorr = max(corr(:))
[yPeak, xPeak] = find(corr == maxCorr)
yOffset = yPeak - size(cropped, 1);
xOffset = xPeak - size(cropped, 2);
result = insertMarker(Ing, [xOffset, yOffset])
imshow(result)
```

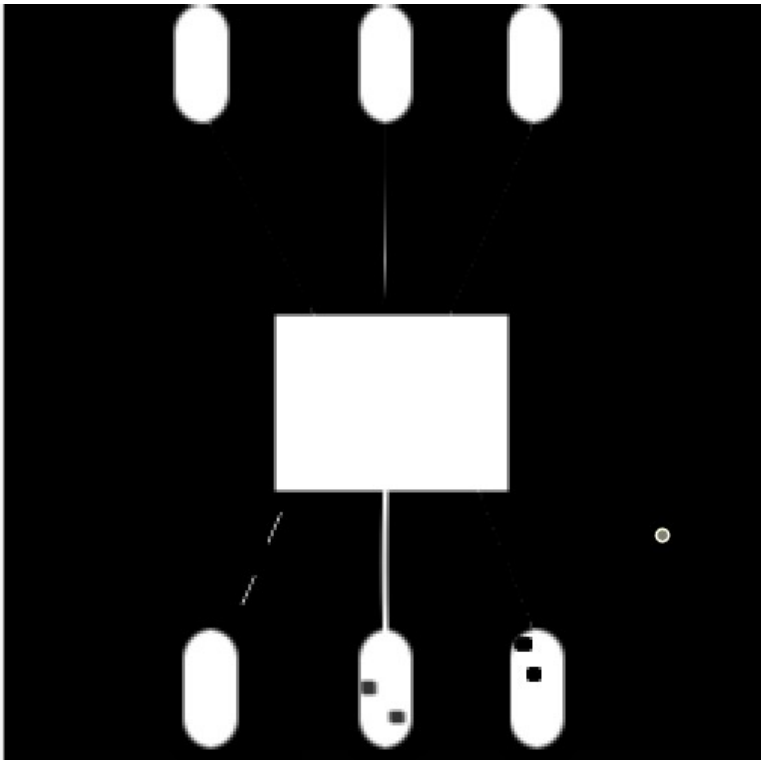
Le max de la corrélation correspond au coin haut gauche du crop.

14) se = [1,1,1;1,1,1;1,1,1]
I_dil = imdilate(I, se);
imshow(I_dil)



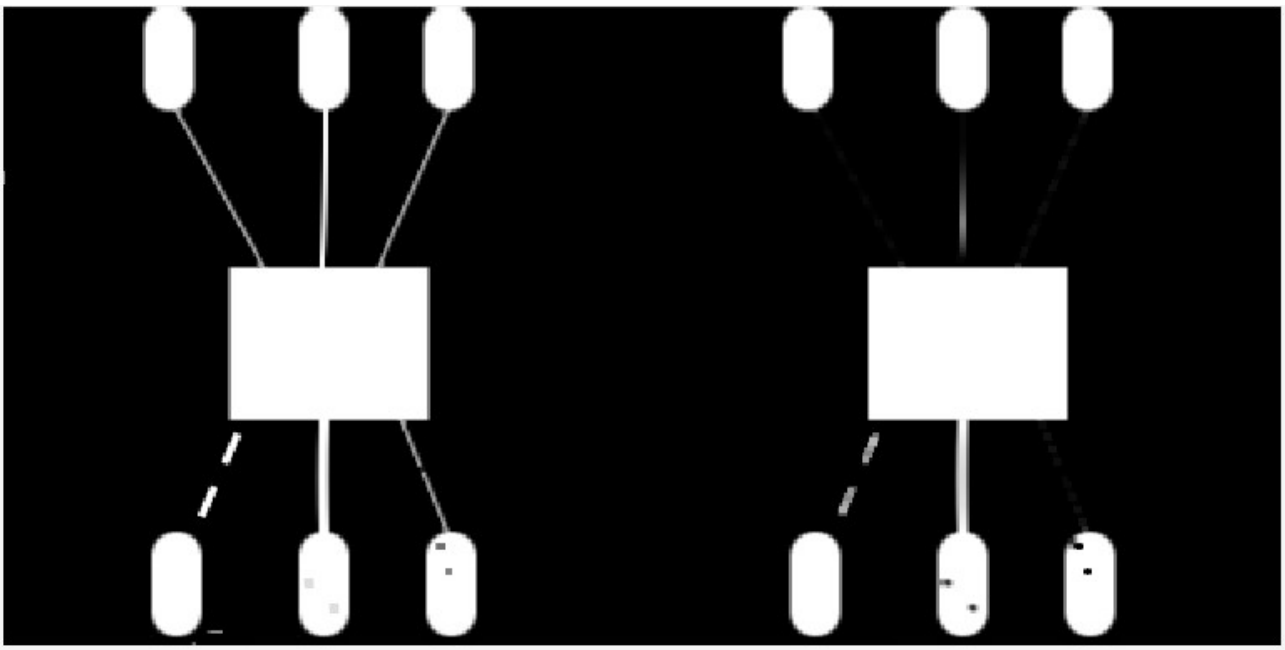
On constate que les trous en bas à droite sont encore présents, mais plus petits. On pourrait peut-être les fermer totalement avec un élément structurant plus fort.

15) I_ero = imerode(I, se);
imshow(I_ero)



On constate que les traits blancs s’effacent et que les trous s’agrandissent.

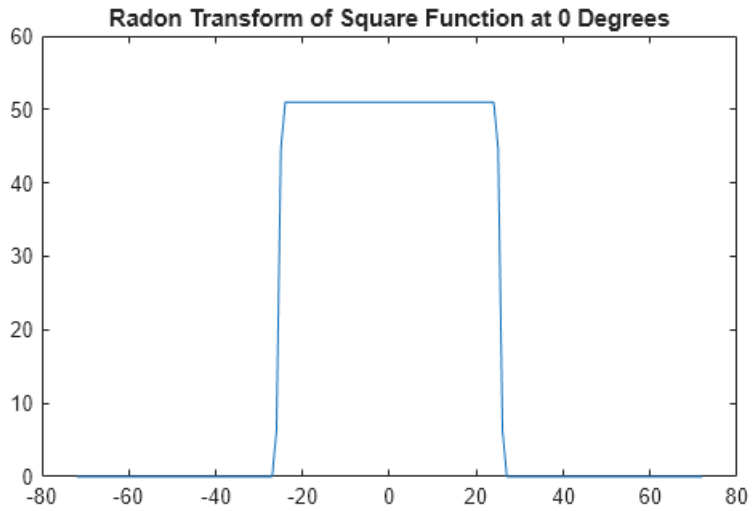
16) I_dil_ero = imerode(I_dil, se)
I_ero_dil = imdilate(I_ero, se)



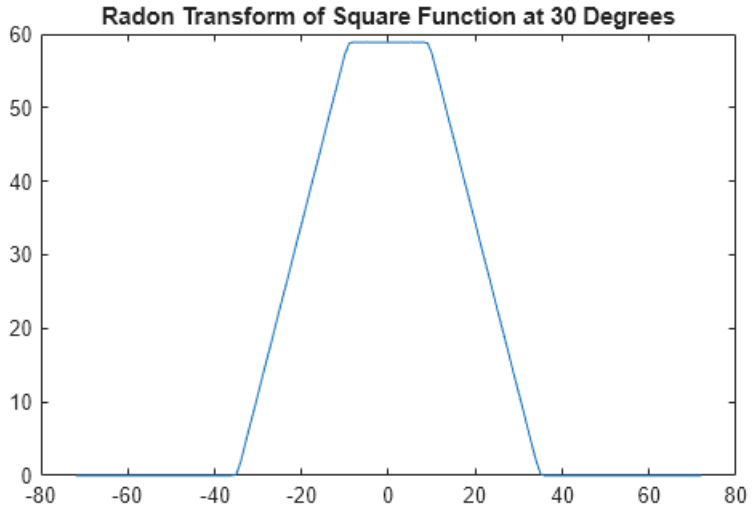
17) I = zeros(100,100);
I(25:75,25:75) = 1;
imshow(I)

theta = [0 30];
[R,xp] = radon(I,theta);

figure
plot(xp,R(:,1))
title("Radon Transform of Square Function at 0 Degrees")

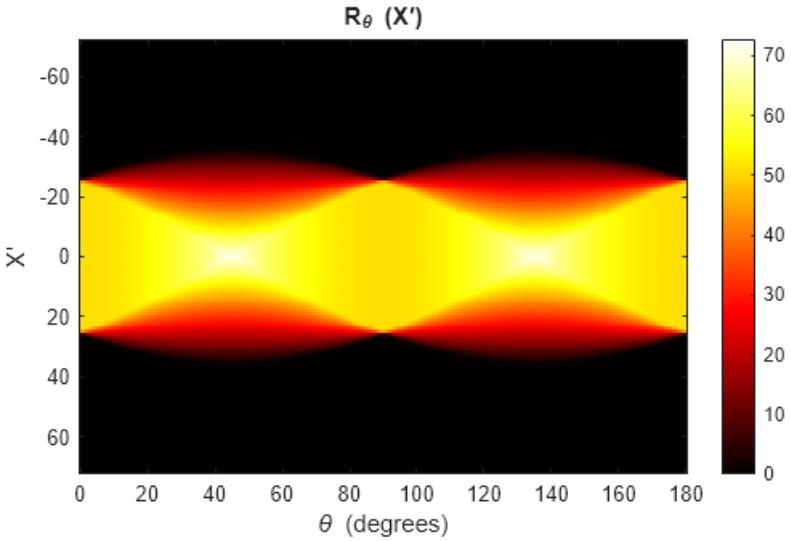


```
plot(xp,R(:,2));
title("Radon Transform of Square Function at 30 Degrees")
```




```
theta = 0:180;
[R,xp] = radon(I,theta);

figure
imagesc(theta,xp,R)
title("R_{\theta} (X\prime)")
xlabel("\theta (degrees)")
ylabel("X\prime")
set(gca,"XTick",0:20:180)
colormap(hot)
colorbar
```



18)