Transformée(s) de Hough - Espaces de votes

I. Modélisation de segments de droite

Image de départ

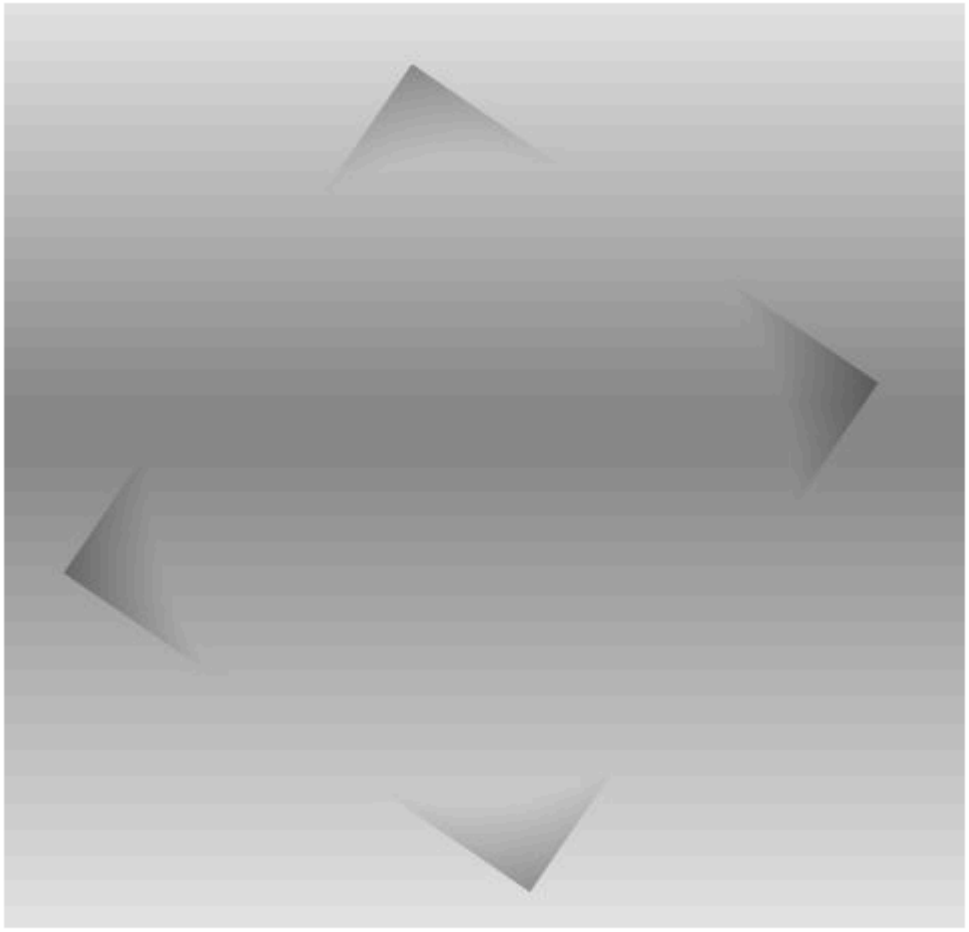
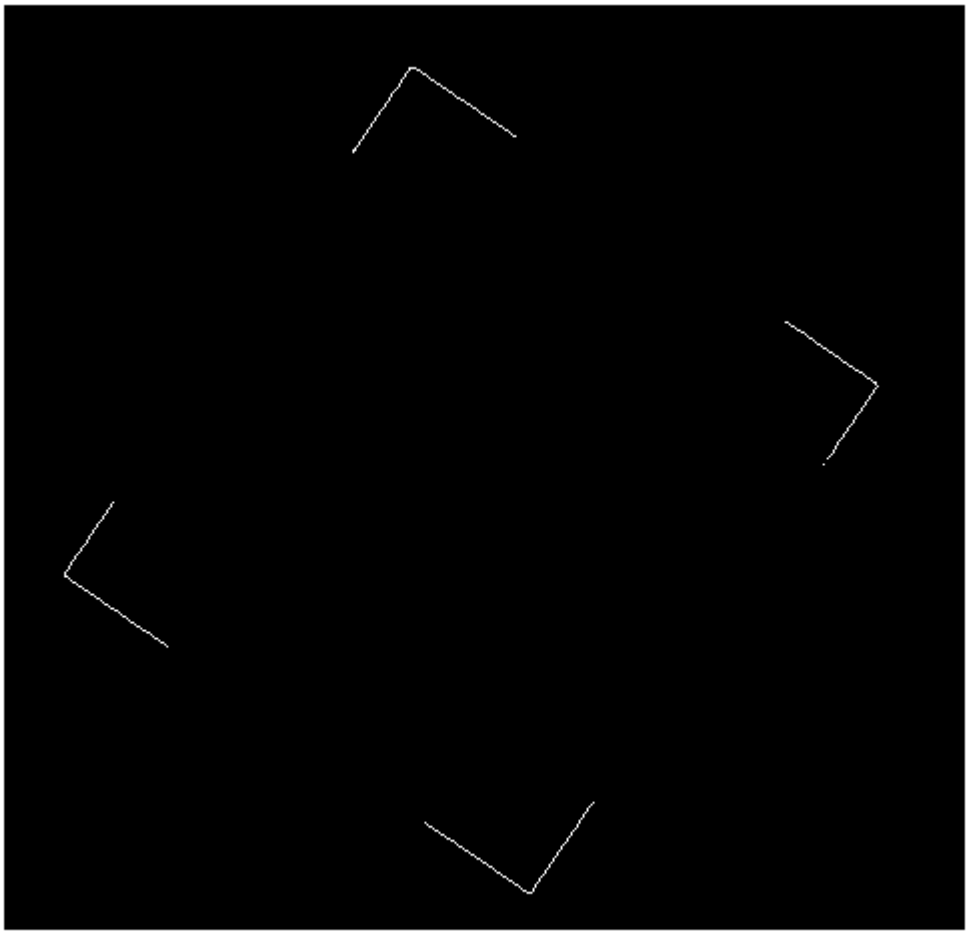
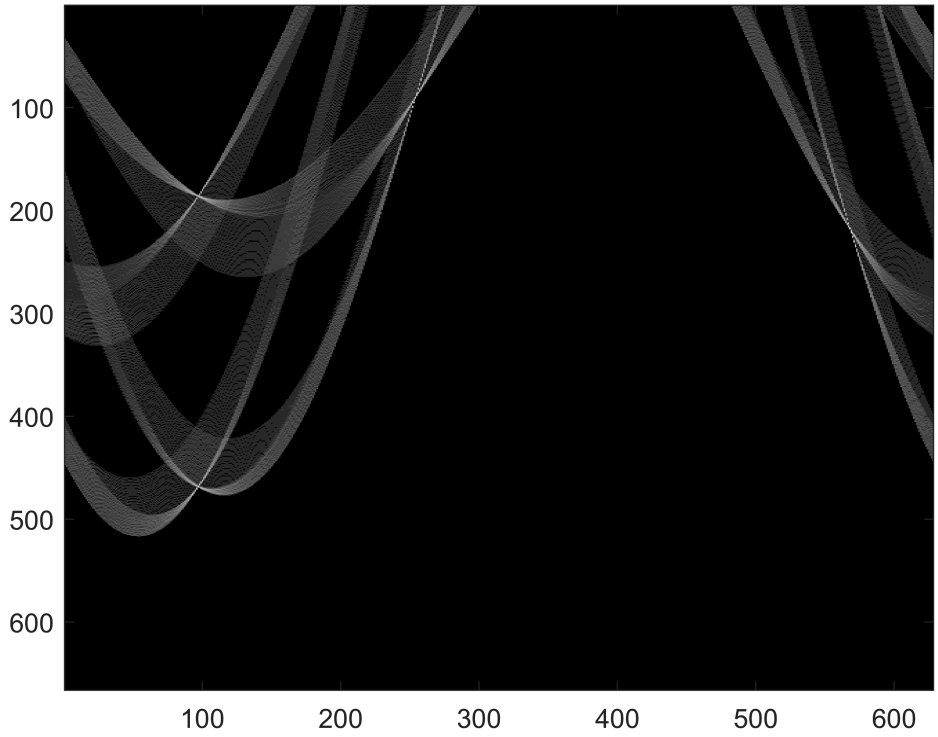


Image binarisé



Espace de vote



II. Reconnaissance de segments de droite

Liste_max = [x y valeur]

Pour k=1

```
Liste_max = 1×3
   186    97    76
```

Pour k=3

```
Liste_max = 3×3
   186    97    76
    90   254    63
   217   568    62
```

Pour k=4

```
Liste_max = 4×3
   186    97    76
    90   254    63
   217   568    62
   468    97    51
```

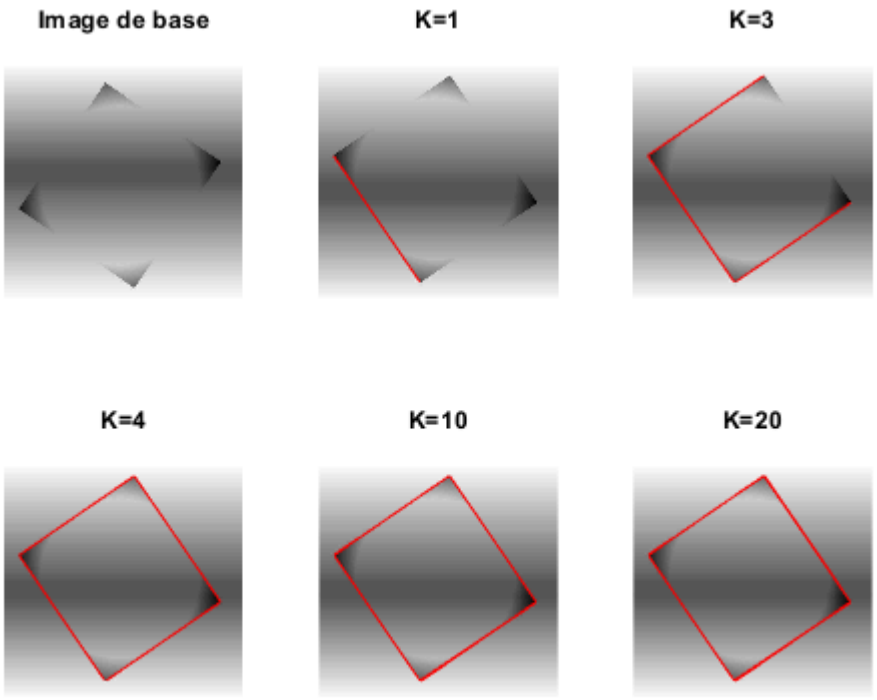
Pour k=10

```
Liste_max = 10×3
   186    97    76
    90   254    63
   217   568    62
```

```
   468    97    51
    88   255    49
   220   569    46
   218   568    43
    95   253    42
    86   256    41
    86   255    40
```

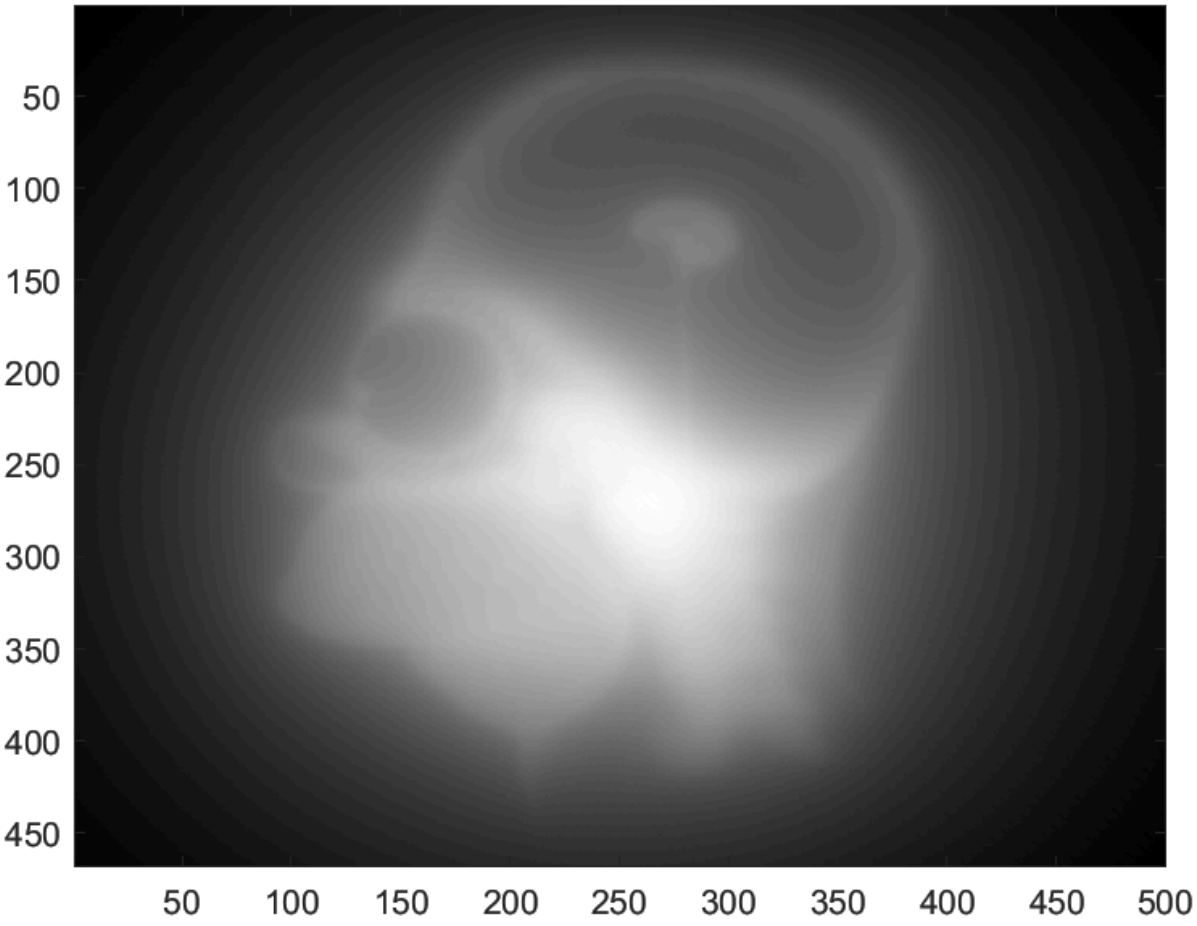
Pour k=20

```
Liste_max = 20×3
   186    97    76
    90   254    63
   217   568    62
   468    97    51
    88   255    49
   220   569    46
   218   568    43
    95   253    42
    86   256    41
    86   255    40
     ⋮
```



III. Back projection of the Radon transform

On obtient l'image suivante en appliquant la rétro-projection de la transformée de Radon sur l'espace de vote fourni



IV Generalized Hough Transform

Je produis le modèle de la forme au sens de la transformée de Hough Généralisé en completant le code proposé et les differentes fonctions

Fonction **contour**:

- listecontour -> liste des points formant le contour
- nbpoints -> taille de listecontour ou nombre de point formant le contour

Fonction **beta** prend en entrée:

- listecontour -> la liste des points d'un contour
- img -> l'image dont est issu le contour
- indice -> l'indice du point du contour a traiter

Fonction **alpha** prend en entrée:

- listecontour -> la liste des points d'un contour
- indice -> l'indice du point du contour a traiter

- img -> l'image dont est issu le contour

```
img=imread('TP03I03.bmp');
img=img(:,:,1)>0;
figure();
subplot(1,2,1)
imshow(img)
[r3,c3]=size(img);
Im_contour=zeros(r3,c3);
% Recovering information about the contour
[C, N]=contour(img);
for i=1:N
    Im_contour(C(i,1),C(i,2))=1;
end
subplot(1,2,2)
imshow(Im_contour);
% Initialize the model :
% row = beta value * 100
% column = order number of the couple (alpha, distance)
% third dimension: for 1 = alpha, for 2 = distance
H=zeros(round(100*2*pi),N,2);
% Calculation of the coordinates of the barycenter
[xo,yo]=barycentre(img);
hold on
scatter(yo,xo,'rx');
title("Contour & baricentre");
hold off
```

```
% Browsing the points on the contour
for i=1:N
    % calculation of b=beta for the ith point of the contour
    b=beta(C,img,i);
    b=round(b);
    % search for the first "empty" column in the table on the line beta
    k=1;
    while H(b+1,k,2)~=0
        k=k+1;
    end
    % record of the value of alpha and the distance on the b+1 row and the kth column of H
    a = alpha(C, i, img);
    d=distance(xo,yo,b,k);
    H(b+1,k,1)=a;
    H(b+1,k,2)=d;
end
```

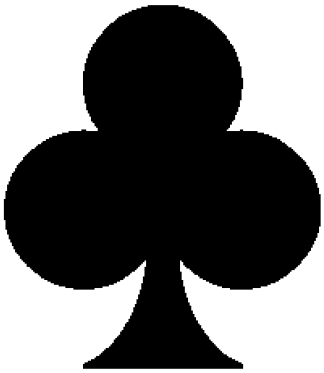
```
function [xo,yo]=barycentre(img)
[x,y]=size(img);
cpt=0;
adx=0;
ady=0;
for i=1:x
    for j=1:y
        if img(i,j)==0
            adx=adx+i;
            ady=ady+j;
            cpt=cpt+1;
        end
    end
end
xo=round(adx/cpt);
yo=round(ady/cpt);
end
```

```
function d=distance(x1,y1,x2,y2)
d = sqrt((x2-x1)^2 + (y2-y1)^2);
end
```

IV.3 Recognition

Je produis l'espace de vote pour les 4 formes ci-dessous

Image binaire



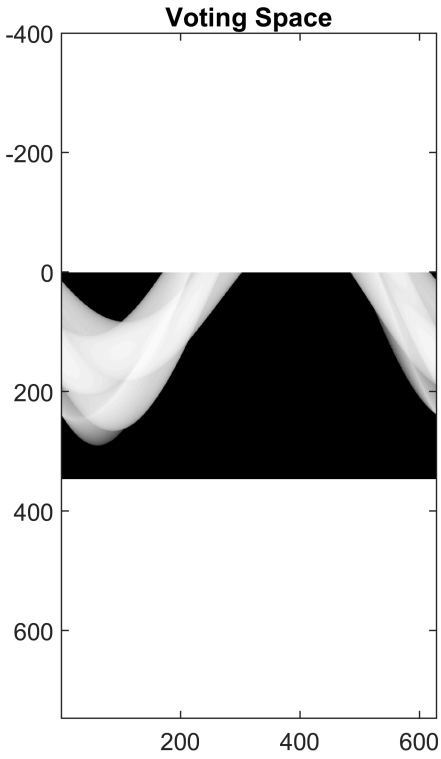
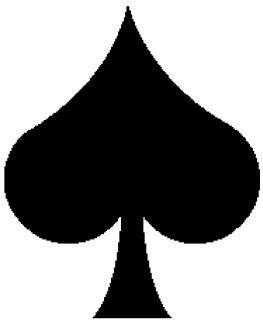


Image binaire



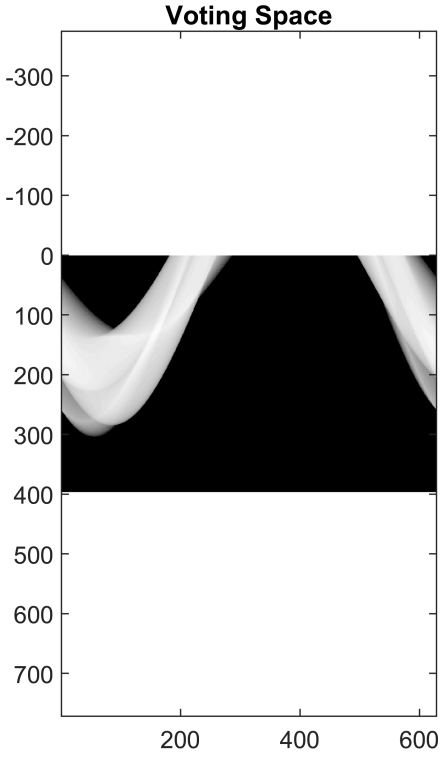


Image binaire

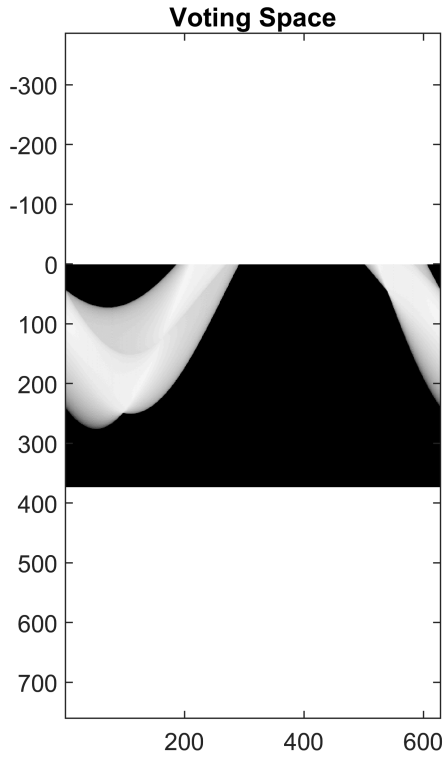


Image binaire