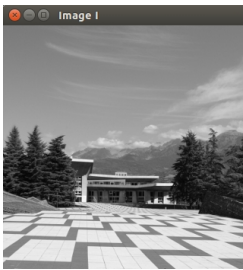
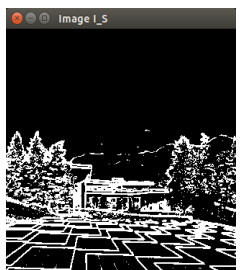









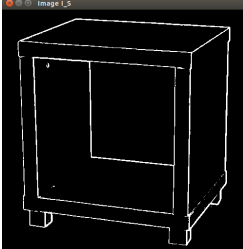
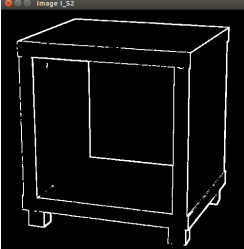
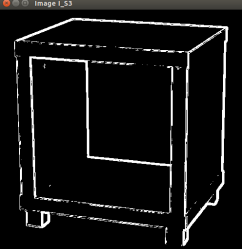
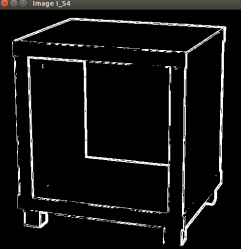


## TP3 : Analyse d'image - détection de contour

### 1 - Filtres détecteurs de contour

#### 1.1 - Traitement d'image en niveaux de gris

Exercice 1 :

Image Initiale	Filtre dérivée méthode 1	Filtre dérivée méthode 2	Filtre Sobel	Filtre Prewitt
				
				
				

On remarque peu de différences entre le filtre dérivée méthode 1 et le filtre dérivée méthode 2 : la méthode 1 reste quand même plus efficace que la deuxième car elle détecte des petits morceaux de contours que l'autre ne détecte pas. De plus, la méthode 2 montre des contours bien moins nettes que la première. Ensuite, les filtres Sobel et Prewitt qui se ressemblent beaucoup, affichent des contours peu nettes et de nombreux vides et oublies. Les contours sont très épais. Par contre il est important de préciser également que Sobel et Prewitt semblent ne pas prendre en compte les pointillées ; ce qui est un point positif pour les images avec bruit !


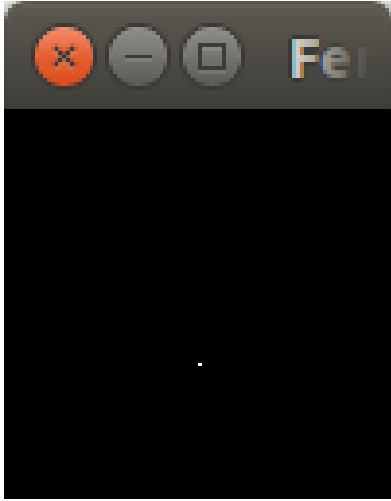
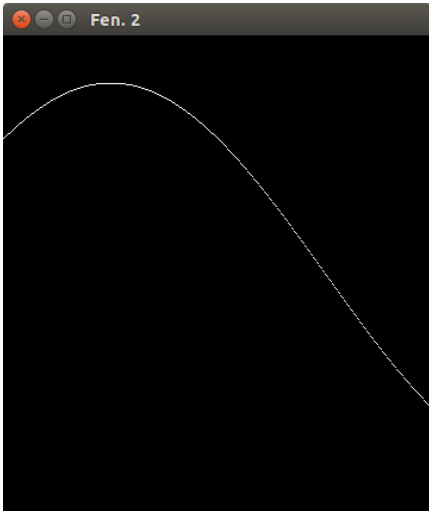
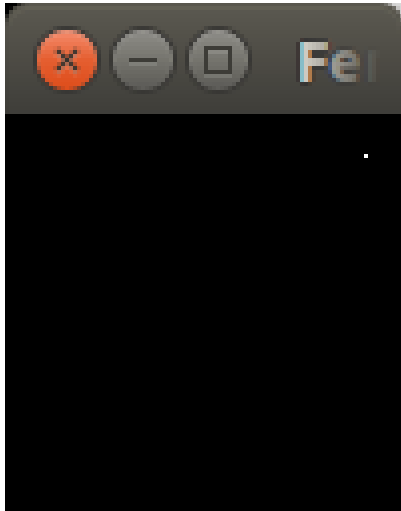
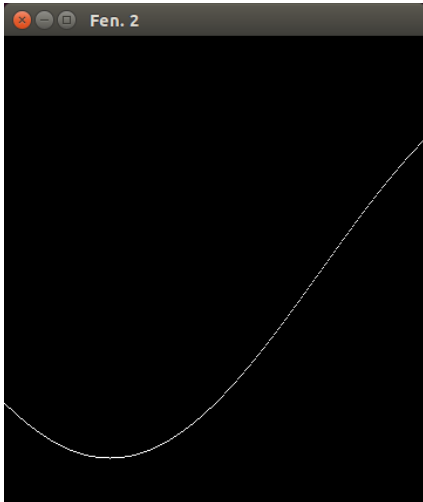
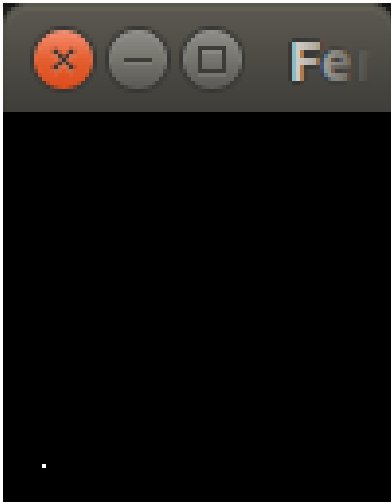
Conclusion : le filtre dérivée méthode 1 reste selon nous la méthode la plus efficace avec un résultat nette et avec le moins d'oublis de contours.

## 2 - Détection de droite par transformée de Hough

### • Partie A

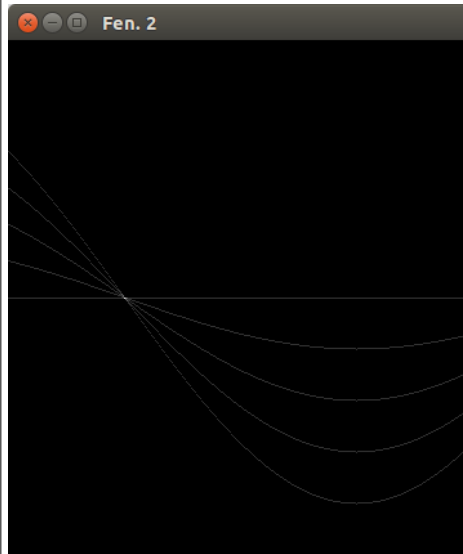
Exercice 2 :

1) Modifiez la position du pixel blanc afin de tester différentes configurations

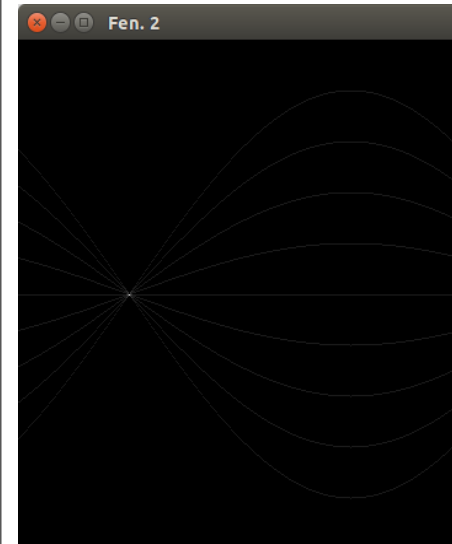
<code>dessiner_pixel(I, 50, 65, 1.0, 1);</code>		
<code>dessiner_pixel(I, 90, 10, 1.0, 1);</code>		
<code>dessiner_pixel(I, 10, 90, 1.0, 1);</code>		

2) Mettez quelques pixels blancs alignés, et testez différentes configurations

```
dessiner_pixel(I, 10, 10, 1.0, 1);  
dessiner_pixel(I, 20, 20, 1.0, 1);  
dessiner_pixel(I, 30, 30, 1.0, 1);  
dessiner_pixel(I, 40, 40, 1.0, 1);  
dessiner_pixel(I, 50, 50, 1.0, 1);
```



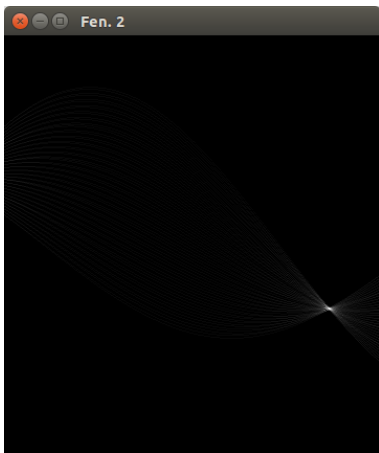
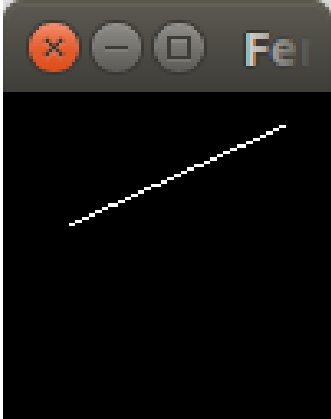
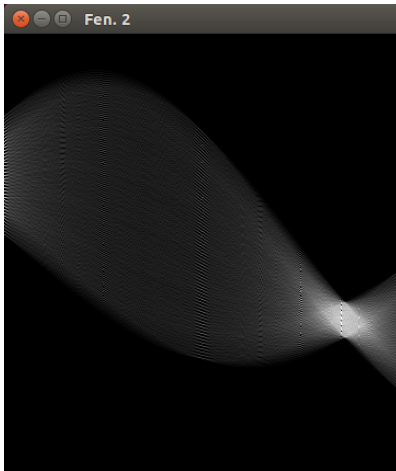
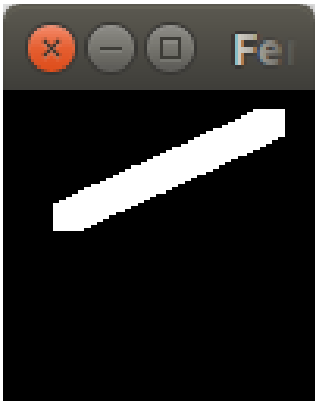
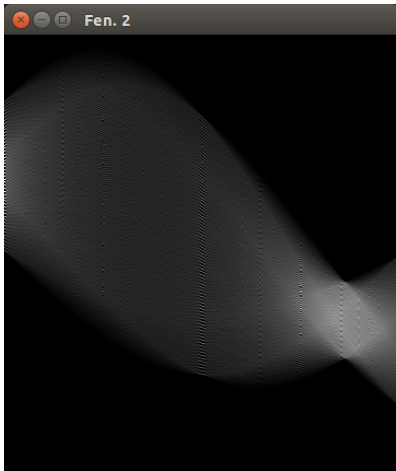
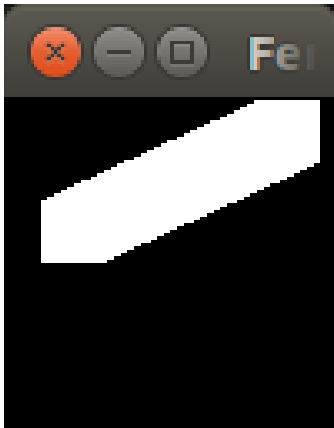
```
dessiner_pixel(I, 10, 10, 1.0, 1);  
dessiner_pixel(I, 20, 20, 1.0, 1);  
dessiner_pixel(I, 30, 30, 1.0, 1);  
dessiner_pixel(I, 40, 40, 1.0, 1);  
dessiner_pixel(I, 50, 50, 1.0, 1);  
dessiner_pixel(I, 60, 60, 1.0, 1);  
dessiner_pixel(I, 70, 70, 1.0, 1);  
dessiner_pixel(I, 80, 80, 1.0, 1);  
dessiner_pixel(I, 90, 90, 1.0, 1);
```



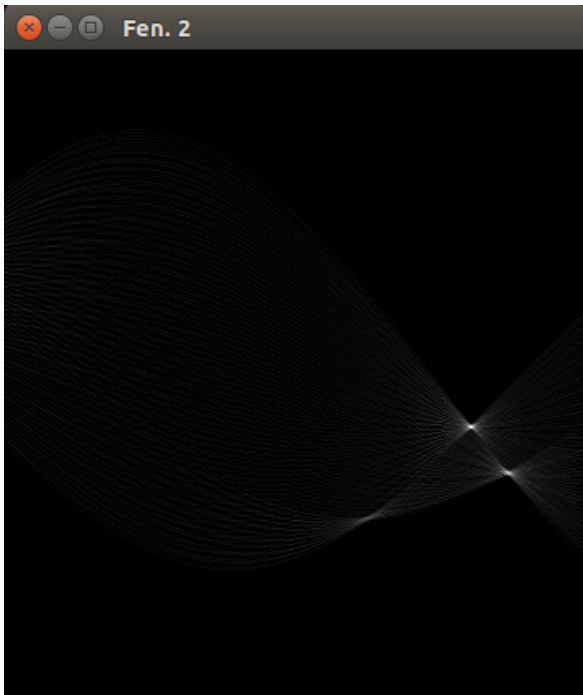
3) Modifiez le segment afin de tester différentes configurations

<code>dessiner_segment(1, 20,40 , 85,10 , 1.0 , 1);</code>		
<code>dessiner_segment(1, 10,10 , 80,80 , 1.0 , 1);</code>		
<code>dessiner_segment(1, 10,65 , 20,40 , 1.0 , 1);</code>		

4) Modifiez l'épaisseur du segment

<code>dessiner_segment(I, 20,40 , 85,10 , 1.0 , 1);</code>		
<code>dessiner_segment(I, 20,40 , 85,10 , 1.0 , 5);</code>		
<code>dessiner_segment(I, 10,65 , 20,40 , 1.0 , 10);</code>		

5) Ajoutez d'autres segments, par exemple pour tracer un triangle



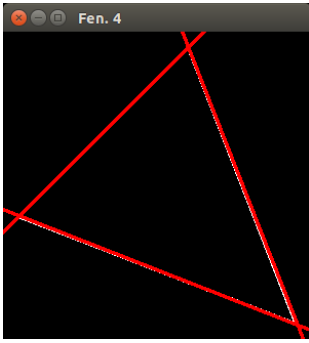
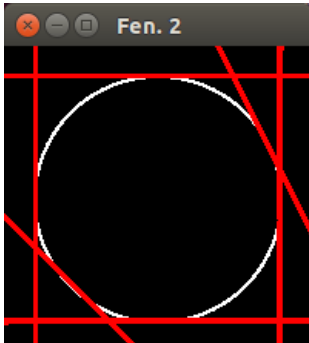
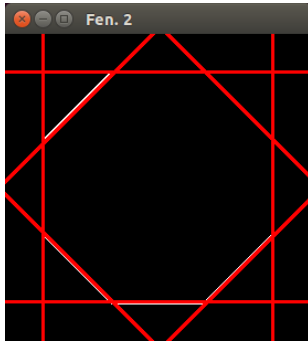
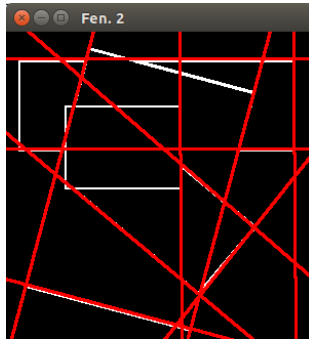
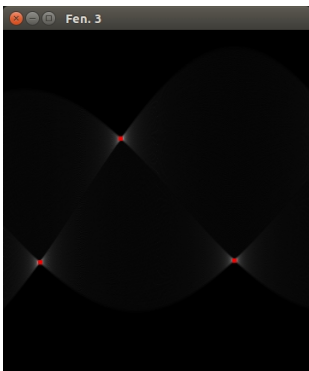
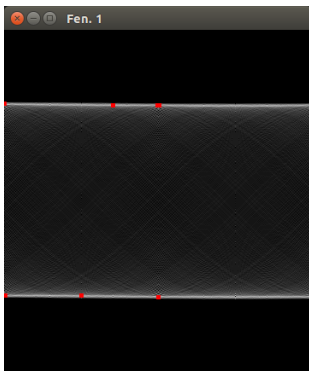
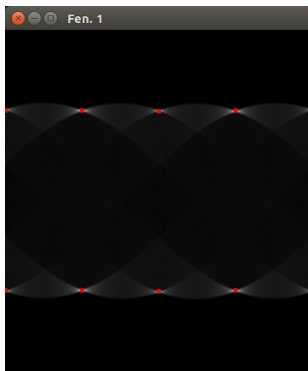
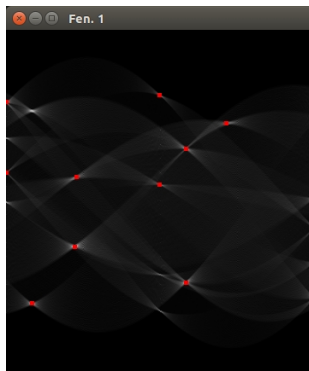
• Partie B

Exercice 3 :

Avant modification :

contour-triangle.png	contour-cercle.png	contour-octogone.png	contour-rectangles.png

Après modification :

<p>contour-triangle.png avec les paramètres de base : seuil = 1.0/3.0; d = 1;</p>	<p>contour-cercle.png avec seuil = 1.0/2.0; d = 30; (ce n'est pas optimal)</p>	<p>contour-octogone.png avec seuil = 1.0/6.0; d = 10; (c'est optimal)</p>	<p>contour-rectangles.png avec seuil = 1.0/10.0; d = 38; (ce n'est pas optimal)</p>
			
			

### 3 - Traitement complet

#### Exercice 4 :

Dans cet exercice et en faisant le lien avec l'exercice précédent, nous avons pu remarquer que dans une image de paysage par exemple, étant donnée la complexité des formes il est difficile de trouver une valeur de seuil et de distance qui puisse détecter toutes les lignes droites possibles. Cependant, avec les images de formes que vous nous avez fournis, il est possible d'avoir un résultat satisfaisant.

De plus, une image de meilleur qualité avec une résolution élevée donnera dans ce programme un résultat prometteur. Nous avons choisi le filtre dérivée méthode 1 car c'était selon nous le filtre le plus efficace avec l'image que nous avons choisi.

Testons avec l'image rubik-cube.png :

Image Initiale





Image convertis en niveaux de gris



Image avec le filtre dérivée methode1  
et un seuillage

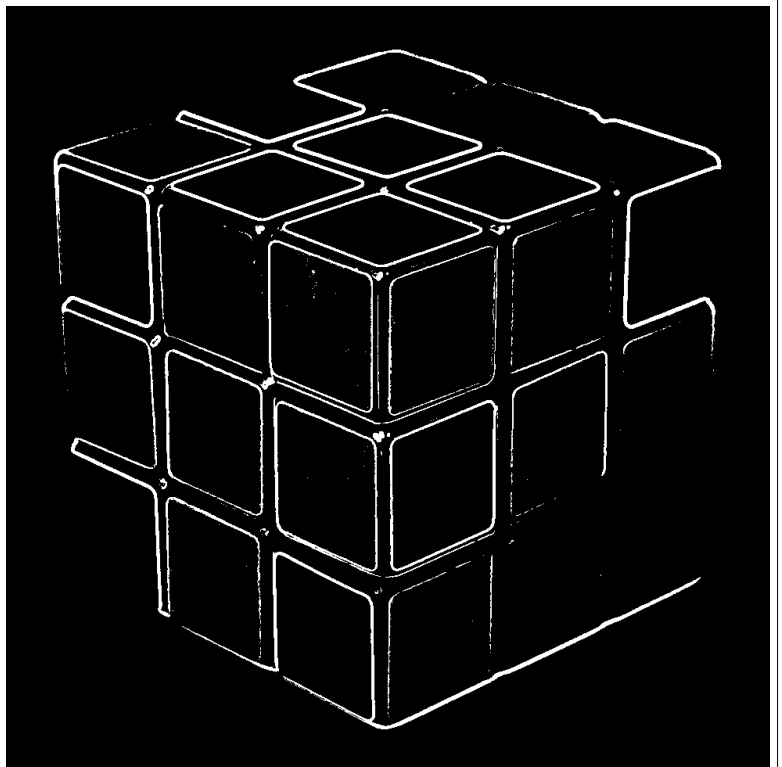


Image avec détection des droites

