

Détection de ballon de rugby dans une image

MONTEIL Clara

18272

2023/2024

Présentation

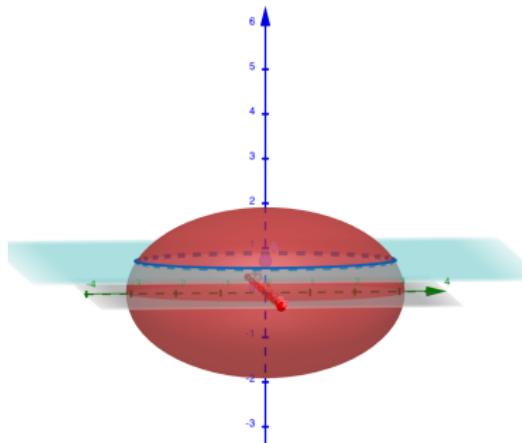
Problématique

Comment modéliser le ballon de rugby et le reconnaître informatiquement dans une image ?

Objectifs et Plan

- Modéliser un ballon de rugby
- Implémenter le filtre de Canny pour détecter les contours
- Reconnaître le ballon à l'aide de la transformée généralisée de Hough

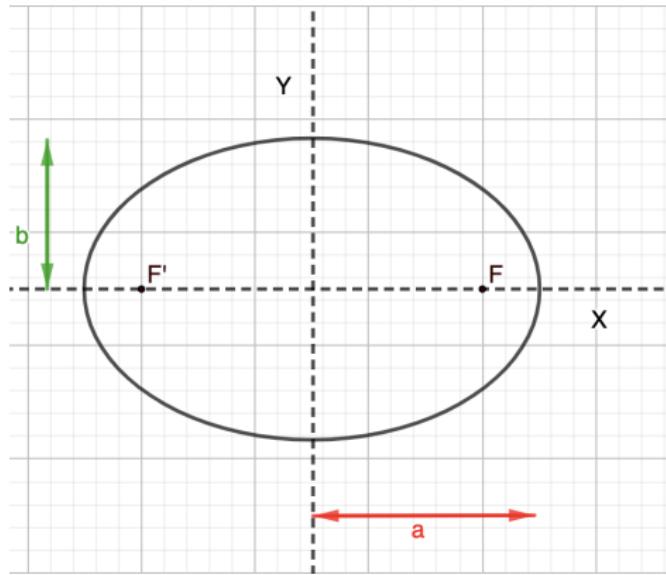
L'ellipse pour représenter le ballon



L'ellipse pour représenter le ballon

Equation

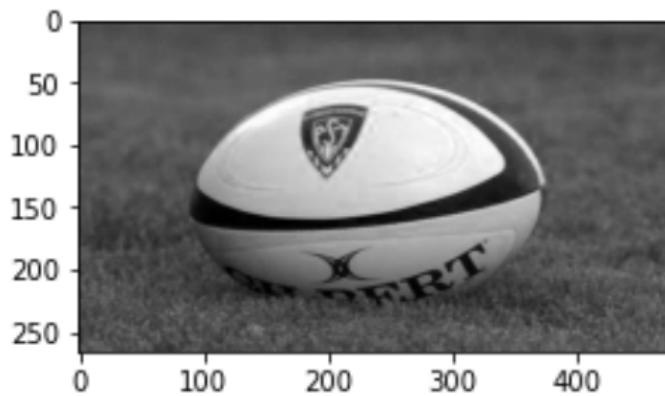
$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$



Traitement de l'image

Image en niveau de gris et atténuation du bruit

- on attribue à chaque pixel son niveau de gris pour pouvoir travailler avec la fonction intensité
- on diminue le bruit avec un filtre (flou gaussien)



Traitement de l'image : Filtre de Canny

Implémentation du filtre de Canny pour détecter les contours

Etape 1 : calcul du gradient de l'image

Etape 2 : suppression des non maxima

Etape 3 : seuillage

Filtre de Canny : Etape 1

Filtre de Sobel pour approximer le gradient

$$S_x = 1/8 \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad S_y = 1/8 \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Calcul du gradient de l'image

Amplitude : $\|\nabla(I(x, y))\| = \sqrt{(S_x * I(x, y))^2 + (S_y * I(x, y))^2}$

Direction : $\Theta = \arctan \left(\frac{S_x * I(x, y)}{S_y * I(x, y)} \right)$

Filtre de Canny

Etape 2 : Suppression des non-maxima

Pour rendre les bord plus précis on ne garde que les points pour lesquels l'amplitude du gradient est localement maximale dans sa direction.

Etape 3 : Seuillage

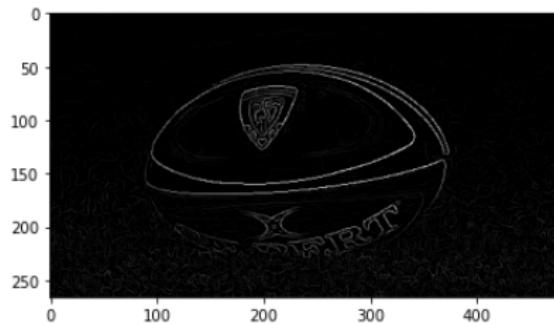
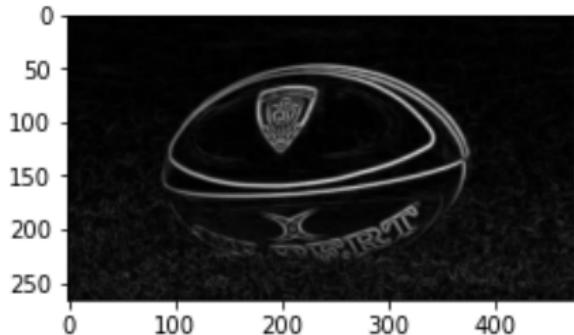
Finalement on ne retient que les points dont l'amplitude est supérieure à un certain seuil. On peut choisir le seuillage par hystérésis : on définit deux seuils s_{bas} et s_{haut} .

- si l'amplitude du gradient $> s_{haut}$, alors le pixel (x,y) est un bord
- si l'amplitude du gradient $> s_{bas}$, alors le pixel (x,y) est situé sur un bord si l'un de ses huit voisins l'est aussi.

Résultats du filtre de Canny

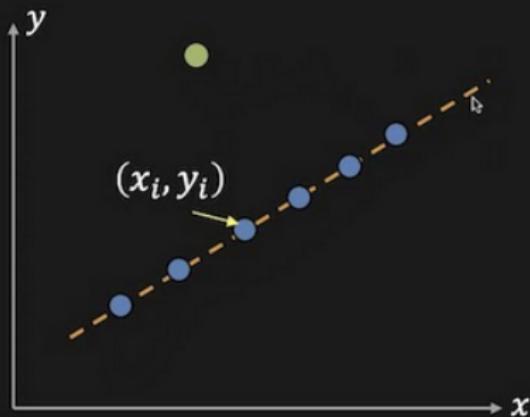
Influence du seuillage

Ballon avant et après seuillage



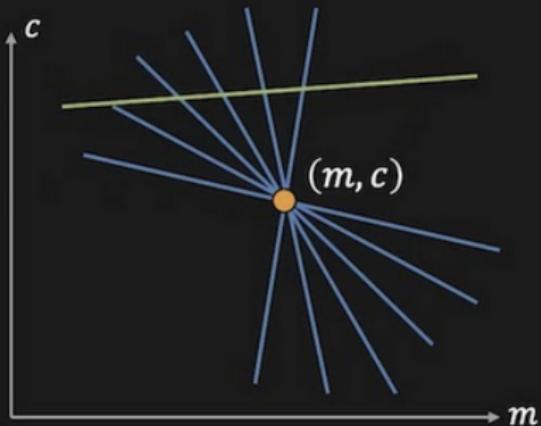
Transformée de Hough

Image Space



$$y_i = mx_i + c$$

Parameter Space

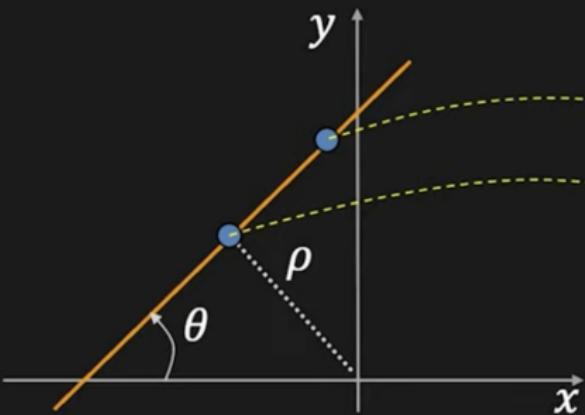


$$c = -mx_i + y_i$$

Schree K.Nayar, University of Columbia

Transformée de Hough

Image Space



$$x \sin \theta - y \cos \theta + \rho = 0$$

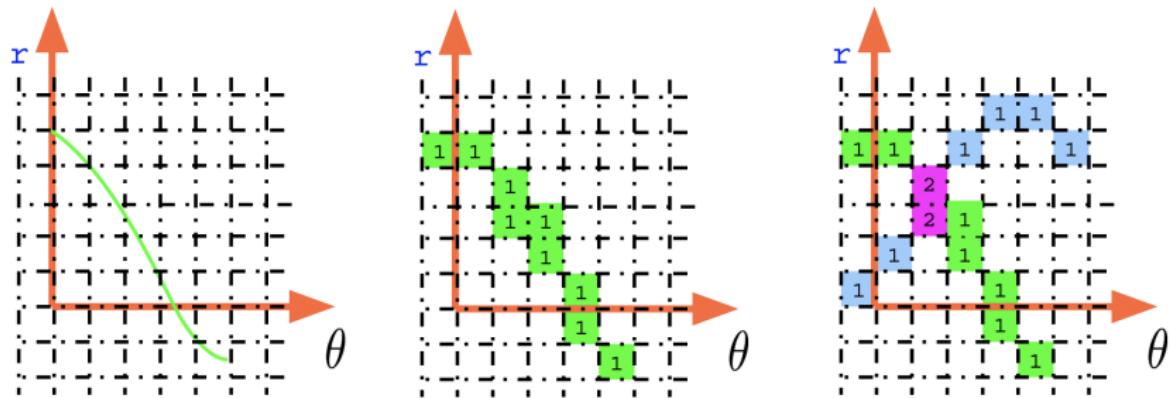
Parameter Space



$$x \sin \theta - y \cos \theta + \rho = 0$$

Schree K.Nayar, University of Columbia

Transformée de Hough



Bruno Vallet, ENSG

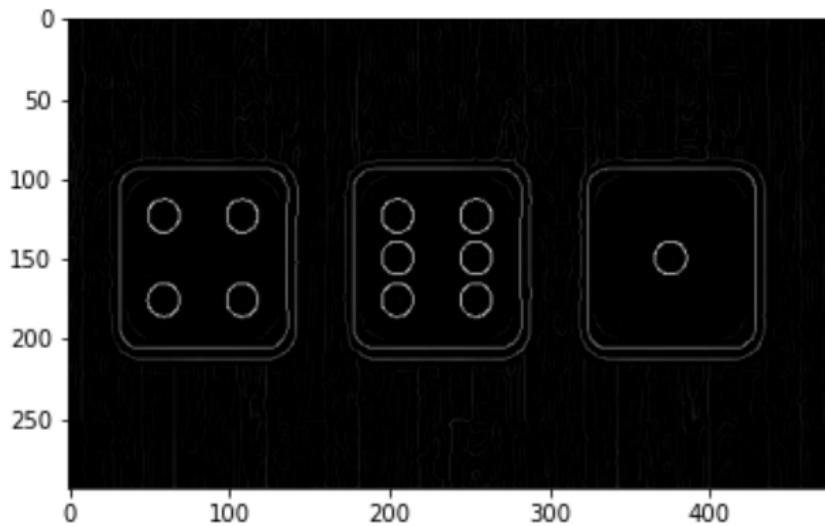
Première approche : droites

Choix de l'image

On a choisi de travailler avec une image peu complexe qui contient plusieurs droites perpendiculaires plus ou moins visibles.



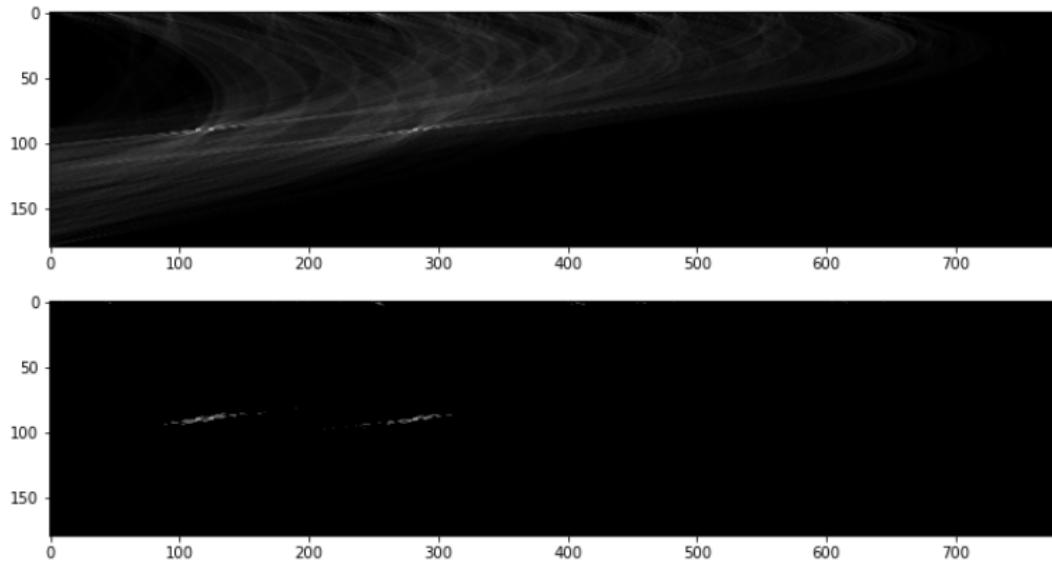
Première approche : droites



Première approche : droites

Résultats

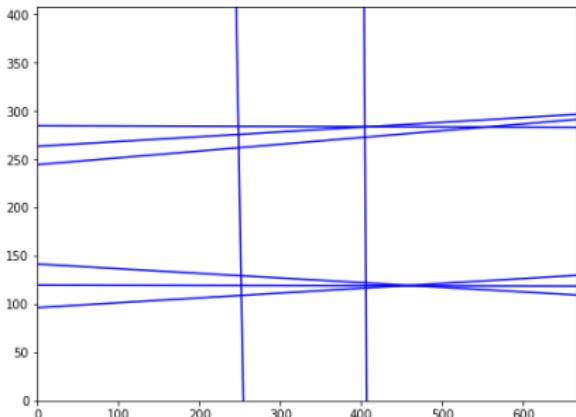
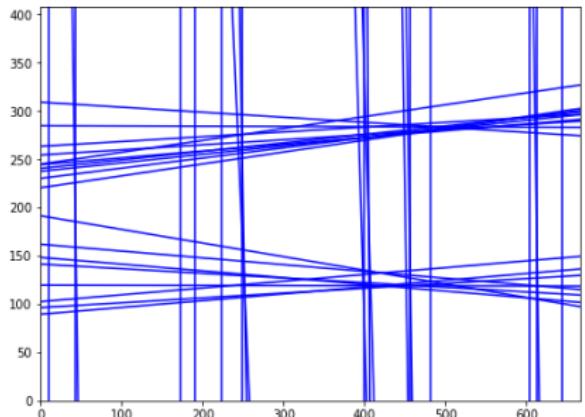
Matrices accumulatrices avant et après seuillage



Transformée de Hough pour les droites

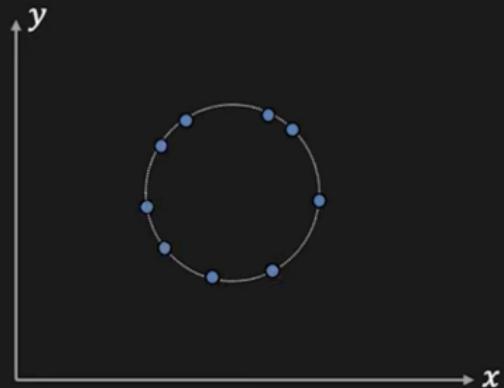
Résultats

Droites détectées avant et après seuillage



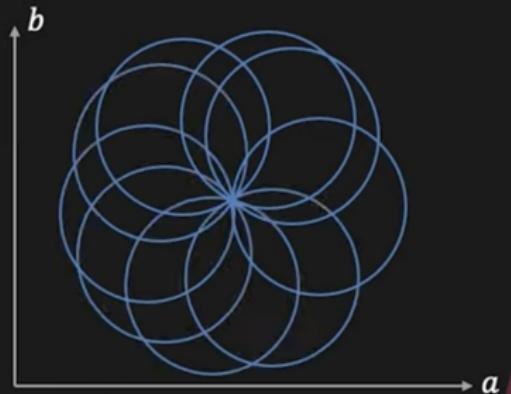
Transformée de Hough

Image Space



$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

Parameter Space



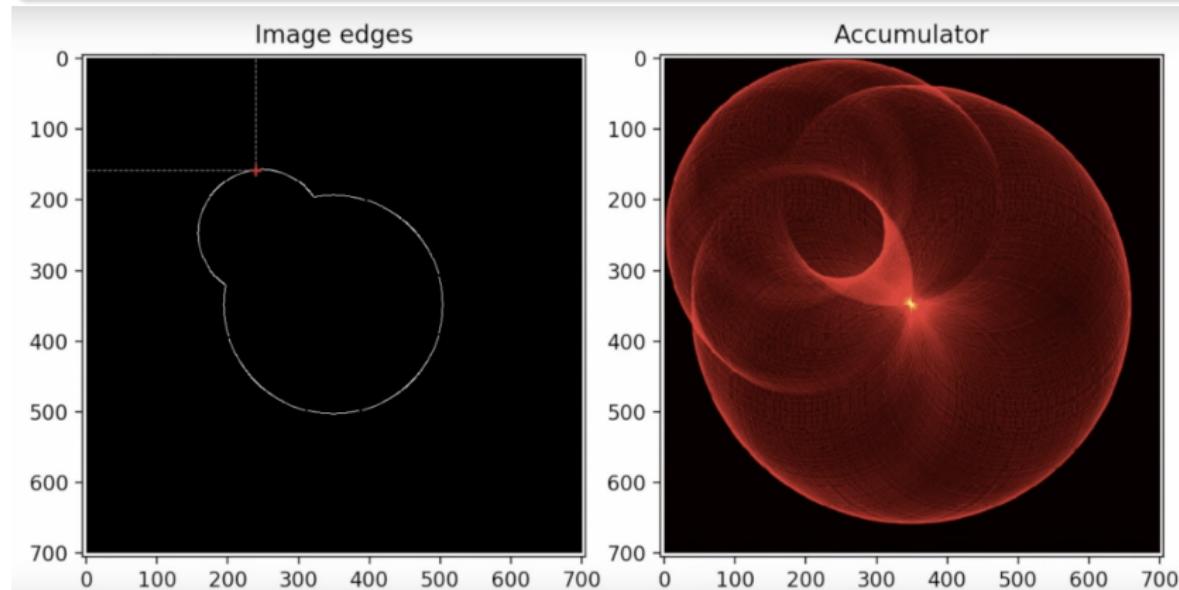
$$(a - x_i)^2 + (b - y_i)^2 = r^2$$

Schree K.Nayar, University of Columbia

Exemple

Problèmes des 3 paramètres

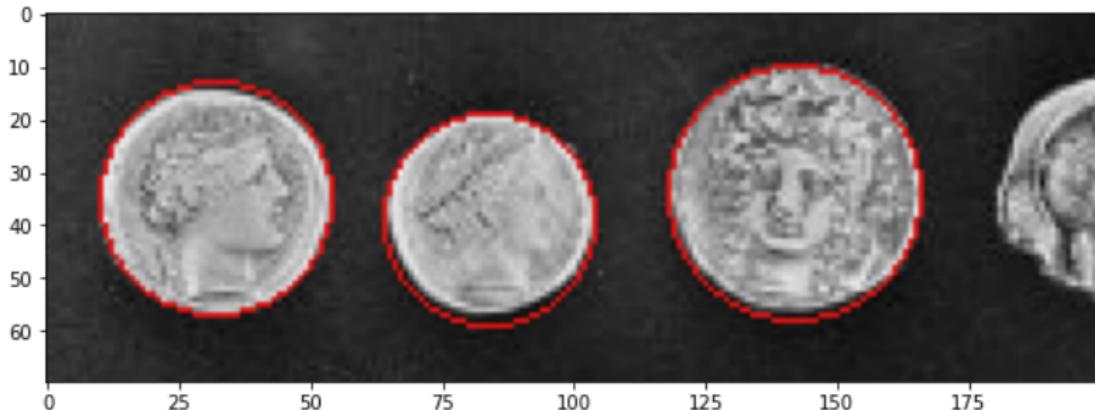
On décide alors de fixer le rayon R du cercle



Exemple

Avec un intervalle pour r

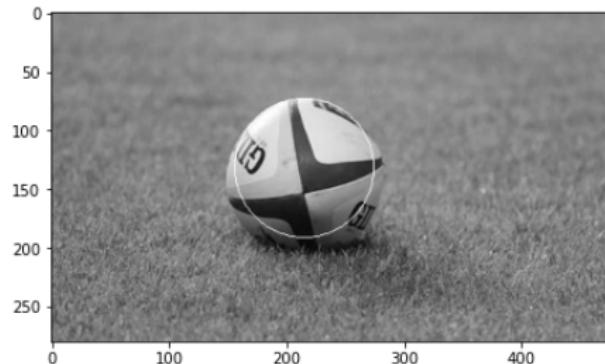
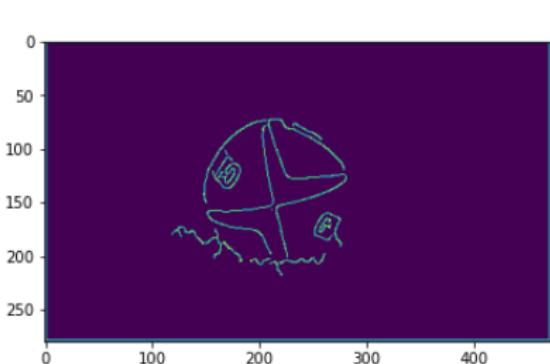
On fait alors varier le rayon



Exemple

Résultat avec l'image de travail

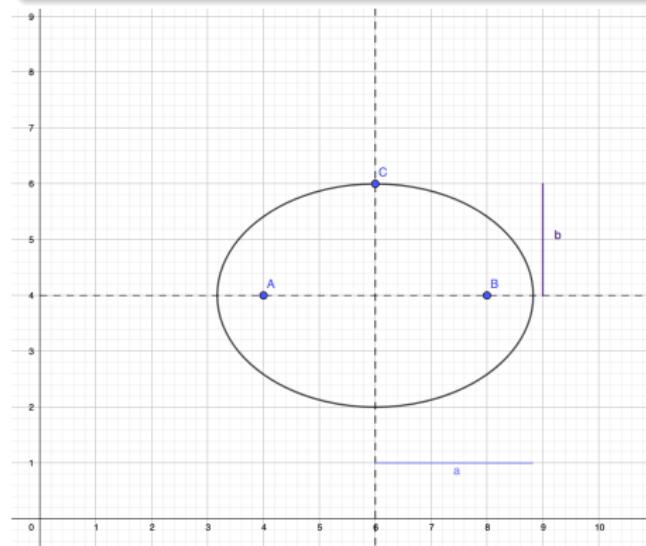
On utilise le filtre de Canny du module scikit-image



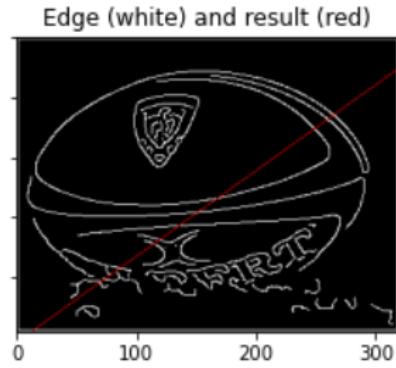
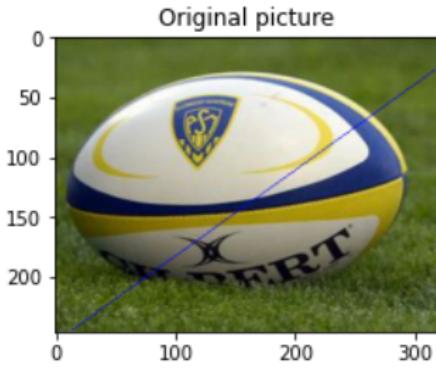
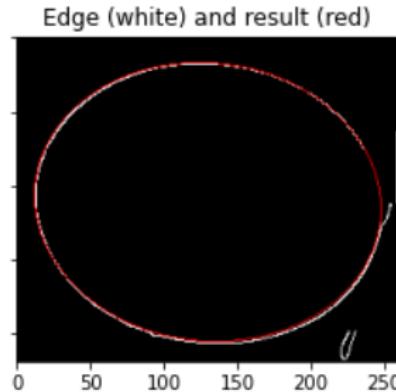
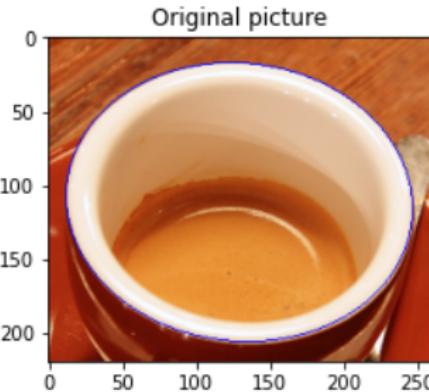
Transformée de Hough ellipse

Problème des 5 paramètres

- demi grand axe a
- demi petit axe b
- centre (x_0, y_0)
- inclinaison θ



Transformée de Hough ellipse



Programme Python : filtre de Canny

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Tue Dec 19 10:16:42 2023
5
6 @author: clara
7
8 #test Canny
9
10
11
12
13 import numpy
14 import math
15 import cmath
16
17 from scipy.ndimage.filters import convolve,gaussian_filter
18 from matplotlib.pyplot import *
19
20
21 matplotlib.pyplot.imread
22 img = imread("ballonrgb.jpg")
23 red = img[:, :, 0]
24 green = img[:, :, 1]
25 blue = img[:, :, 2]
26 array=red*1.0
27 figure(figsize=(4,4))
28 imshow(array,cmap=cm.gray) #IMAGE 1
29
30
31 array = gaussian_filter(array,1)
32 sobelX = numpy.array([-1,0,1],[-2,0,2],[-1,0,1])
33 sobely = numpy.array([-1,-2,-1],[0,0,0],[1,2,1])
34 derivX = convolve(array,sobelX)
35 derivY = convolve(array,sobely)
36 gradient = derivX+derivY*1j
37 G = numpy.absolute(gradient)
38 theta = numpy.angle(gradient)
39
40 figure(figsize=(8,4))
41 f,(p1,p2)=subplots(ncols=2)
42 p1.imshow(derivX,cmap=cm.gray)
43 p2.imshow(derivY,cmap=cm.gray)
44
45 figure(figsize=(4,4))
46 imshow(G,cmap=cm.gray) #IMAGE 2
47
48 seuil = 0.23
49 s = G.shape
50 for i in range(s[0]):
51     for j in range(s[1]):
52         if G[i][j]<seuil:
53             G[i][j] = 0.0
54 figure(figsize=(4,4))
55 imshow(G,cmap=cm.gray) #IMAGE 3
56
```

Programme Python : filtre de Canny

```
65 # SUPPRESSION DES NON MAXIMA
66
67 #copie de la matrice pour retirer les maxima
68 Gmax = G.copy()
69
70 #fonction d'interpolation
71 def interpolation(array,x,y):
72     s = array.shape
73     i = math.floor(x)
74     j = math.floor(y)
75     t = x-i
76     u = y-j
77     u1 = 1.0-u
78     t1 = 1.0-t
79     if j==s[0]-1:
80         if i==s[1]-1:
81             return array[j][i]
82         return t*tarray[j][i]+t1*u*array[j+1][i]
83     if i==s[1]-1:
84         return u*uarray[j][i]+u1*u*array[j][i+1]
85     return t1*u1*uarray[j][i]+t1*t1*u*array[j][i+1]+\
86         t*u*uarray[j+1][i+1]+t1*u*uarray[j+1][i]
87
88 #Suppression des non maxima
89
90 for i in range(1,s[1]-1):
91     for j in range(1,s[0]-1):
92         if G[j][i]==0:
93             cos = math.cos(theta[j][i])
94             sin = math.sin(theta[j][i])
95             g1 = interpolation(G,i*cos,j+sin)
96             g2 = interpolation(G,i*cos,j-sin)
97             if (G[j][i]<g1) or (G[j][i]<g2):
98                 Gmax[j][i] = 0.0
99
100 figure(figsize=(6,6))
101 imshow(Gmax,cmap=cm.gray) #IMAGE 4
102
103 # Seuillage
104
105 Gfinal = Gmax.copy()
106 Gfinal_2 = Gmax_2.copy()
107 seuil = 0.2
108 for j in range(s[0]):
109     for i in range(s[1]):
110         if Gfinal[j][i]<seuil:
111             Gfinal[j][i] = 0.0
112         else:
113             Gfinal[j][i] = 255.0
114         if Gfinal_2[j][i]<seuil:
115             Gfinal_2[j][i] = 0.0
116         else:
117             Gfinal_2[j][i] = 255.0
118 figure(figsize=(10,6))
119 f,(p1,p2)=subplots(ncols=2)
120 p1.imshow(Gfinal,cmap=cm.gray) #IMAGE 5
```

Programme Python : Hough droites

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 Created on Tue Jan 30 11:03:31 2024
5
6 @author: clara
7
8
9 #TRANSFORMEE DE HOUGH - DROITE
10
11
12 import numpy
13 import math
14 import cmath
15 from matplotlib.pyplot import*
16 from PIL import Image
17 img = Image.open("D1.png") # le contraste de l'image a une importance
18 #conséquente sur le résultat par la transformée
19 img = img.convert('L')
20 array=numpy.array(img)
21 figure(figsize=(8,6))
22 img.show()
23 imshow(array,cmap=cm.gray)
24
25 #Test taille de l'image
26
27
28 print(img.size)
29 print(type(img.size))
30
31
32 w, h = img.size
33 print('width: ', w)
34 print('height: ', h)
35
36
37
38
39
40 (Ny,Nx) = array.shape
41 rho= 1.0
42 theta=1.0
43 Ntheta = int(180.0/theta)
44 Nrho = int(math.floor(math.sqrt(Nx*Nx+Ny*Ny))/rho)
45 dtheta = math.pi/Ntheta
46 drho = math.floor(math.sqrt(Nx*Nx+Ny*Ny))/Nrho
47 accum = numpy.zeros((Ntheta,Nrho))
48
```

Programme Python : Hough droites

```
49 #MODIFICATION DES PARAMETRES
50
51 for j in range(Ny):
52     for i in range(Nx):
53         print(array[j][i])
54         if array[j][i] > 10: #le choix du seuil dépend du contraste choisi
55             #pour le contour
56             for i_theta in range(Ntheta):
57                 theta = i_theta+dtheta
58                 rho = i*math.cos(theta)+(Ny-j)*math.sin(theta)
59                 i_rho = int(rho/drho)
60                 if (i_rho>0) and (i_rho<Nrho):
61                     accum[i_theta][i_rho] += 1
62
63
64 figure(figsize=(12,6))
65 print("matrice accumulatrice non seuillée")
66 imshow(accum,cmap=cm.gray)
67 """PREMIERE IMAGE"""
68
69
70 # ok vérification
71
72
73 seuil=130 #c'est un choix arbitraire
74 accum_seuil = accum.copy()
75 for i_theta in range(Ntheta):
76     for i_rho in range(Nrho):
77         if accum[i_theta][i_rho]<seuil:
78             accum_seuil[i_theta][i_rho] = 0
79
80 figure(figsize=(12,6))
81 imshow(accum_seuil,cmap=cm.gray)
82 """DEUXIEME IMAGE"""
83
84 #paramètres à gerer
85
86
87 lignes = []
88 for i_theta in range(Ntheta):
89     for i_rho in range(Nrho):
90         if accum_seuil[i_theta][i_rho]!=0:
91             lignes.append((i_rho*drho,i_theta+dtheta))
92
93
94
```

Programme Python : Hough droites

```
98
99     figure(figsize=(8,6))
100    axis([0,Nx,0,Ny])
101    for rho,theta in lignes:
102        a = math.cos(theta)
103        b = math.sin(theta)
104        x0 = a*rho
105        y0 = b*rho
106        x1 = int(x0 + 1000*(-b))
107        y1 = int(y0 + 1000*(a))
108        x2 = int(x0 - 1000*(-b))
109        y2 = int(y0 - 1000*(a))
110        plot([x1,x2],[y1,y2],color="b")
111
112
113
114
115 def coloriage_pixel_pile(image,result,shape,seuil,valeur,pile,i,j):
116     image[j][i] = valeur
117     result[j][i] = 255
118     voisins = [(i+1,j),(i-1,j),(i,j-1),(i,j+1)]
119     for pixel in voisins:
120         (k,l) = pixel
121         if k>=0 and k<shape[1] and l>=0 and l<shape[0]:
122             if image[l][k]>seuil:
123                 image[l][k] = valeur
124                 pile.append(pixel)
125
126 def analyse(image,seuil):
127     shape = image.shape
128     Nx = shape[1]
129     Ny = shape[0]
130     compteur = 0
131     positions = []
132     result = numpy.zeros((Ny,Nx),dtype=numpy.uint8)
133     for y in range(Ny):
134         for x in range(Nx):
135             if image[y][x] > seuil:
136                 compteur += 1
137                 pile = [(x,y)]
138                 si = 0
139                 sj = 0
140                 npix = 0
141                 while len(pile)>0:
142                     (i,j) = pile.pop()
143                     si += i
144                     sj += j
145                     npix += 1
146                     coloriage_pixel_pile(image,result,shape,seuil,0,pile,i,
147                                         xb = si*1./npix
148                                         yb = sj*1./npix
149                                         positions.append((xb,yb,npix))
150     print("Nombre de taches : %d"%compteur)
151     positions = sorted(positions, key=lambda positions:-positions[2])
152     for p in positions:
153         print("x=%f, y=%f, npix=%d"%(p[0],p[1],p[2]))
154     return (positions,result)
```

Programme Python : Hough droites

```
156
157 (positions,result) = analyse(accum,130)
158 shape = accum.shape
159 figure(figsize=(6,6))
160 for p in positions:
161     plot([p[0]], [shape[0]-p[1]], 'ob')
162     axis([0,shape[1],0,shape[0]])
163
164
165
166
167
168 figure(figsize=(8,6))
169 axis([0,Nx,0,Ny])
170 for point in positions:
171     i_rho = point[0]
172     i_theta = point[1]
173     rho = i_rho*drho
174     theta = i_theta*dtheta
175     a = math.cos(theta)
176     b = math.sin(theta)
177     x0 = a*rho
178     y0 = b*rho
179     x1 = int(x0 + 1000*(-b))
180     y1 = int(y0 + 1000*(a))
181     x2 = int(x0 - 1000*(-b))
182     y2 = int(y0 - 1000*(a))
183     plot([x1,x2],[y1,y2],color="b")
184
185
186
187
188
189 figure(figsize=(8,6))
190 axis([0,Nx,0,Ny])
191 for point in positions[0:8]:
192     i_rho = point[0]
193     i_theta = point[1]
194     rho = i_rho*drho
195     theta = i_theta*dtheta
196     a = math.cos(theta)
197     b = math.sin(theta)
198     x0 = a*rho
199     y0 = b*rho
200     x1 = int(x0 + 1000*(-b))
201     y1 = int(y0 + 1000*(a))
202     x2 = int(x0 - 1000*(-b))
203     y2 = int(y0 - 1000*(a))
204     plot([x1,x2],[y1,y2],color="b")
205
```

Programme Python : Hough cercles

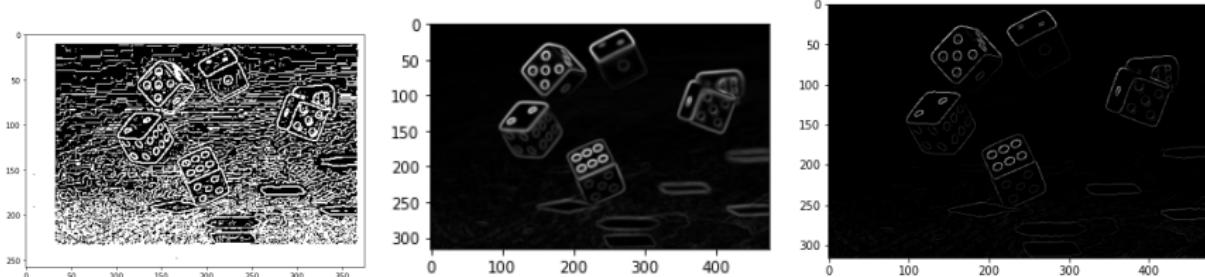
```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Tue Mar 12 10:53:16 2024
5
6 @author: clara
7
8 # TRANSFORMEE DE HOUGH POUR LES CERCLES AVEC LA BIBLIOTHEQUE SCIKIT-IMAGE
9
10
11 import numpy as np
12 import matplotlib.pyplot as plt
13
14 from skimage import data, color
15 from skimage.transform import hough_circle, hough_circle_peaks
16 from skimage.feature import canny
17 from skimage.draw import circle_perimeter
18 from skimage.util import img_as_ubyte
19
20 #IMAGE EN UINT8
21 from skimage.io import imread
22 sk_image = imread("ballonrgbdface.jpg")
23 s_image = sk_image.astype(np.uint8)
24 print(s_image)
25
26
27 #IMAGE EN 2D
28 print(np.shape(s_image))
29
30 image_rgb = s_image[0:500, 1:500]
31 image_gray = color.rgb2gray(image_rgb)
32
33
34 # détection des bords, seuil par hysteresis
35 edges = canny(image_gray, sigma=2.0, low_threshold=0.1, high_threshold=0.3)
36 plt.imshow(edges)
37
38
39
40 # détection du rayon du ballon
41 hough_radii = np.arange(59, 60, 2)
42 hough_res = hough_circle(edges, hough_radii)
43
44 # sélection du rayon prédominant
45 accums, cx, cy, radii = hough_circle_peaks(hough_res, hough_radii,
46 total_num_peaks=1)
47
48 # affichage
49 fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(10, 4))
50 image = color.gray2rgb(image_gray)
51 for center_y, center_x, radius in zip(cy, cx, radii):
52     circy, circx = circle_perimeter(center_y, center_x, radius,
53                                     shape=image.shape)
54     image[circy, circx] = (220, 20, 20)
55
56 ax.imshow(image, cmap=plt.cm.gray)
57 plt.show()
```

Programme Python : Hough ellipse

Autres images obtenues durant l'étude

Problème de seuillage

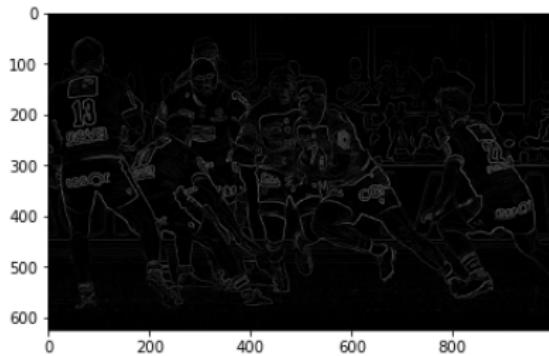
Un mauvais seuillage peut donner une image illisible



Autres images obtenues durant l'étude

Choix de l'image

On s'aperçoit de la difficulté de travailler avec une image trop complexe bien que plus proche de la réalité.



Autres images obtenues durant l'étude

Plusieurs ballon

On s'aperçoit de la difficulté de travailler avec une image trop complexe bien que plus proche de la réalité.

