GOAL LINE TECHNOLOGY

EKOUE-KOUVAHEY

EDEM

SOMMAIRE

1-INTRODUCTION

- Lors d'une situation de jeu au football, les joueurs peuvent effectuer des tirs qui ont une vitesse ahurissante. Dans ces conditions il est beaucoup plus probable de rencontrer de erreurs d'arbitrage ,surtout dues aux limitations humaines , la FIFA a donc décidé d'instaurer la Goal Line Technology : système vidéo d'assistance permettant de déterminer si la ballon a bien franchit la ligne de but.

a-) PROBLEMATIQUE

- Comment pourrais t-on procéder pour localiser la balle

sur le terrain de jeu ?

b-) OBJECTIFS

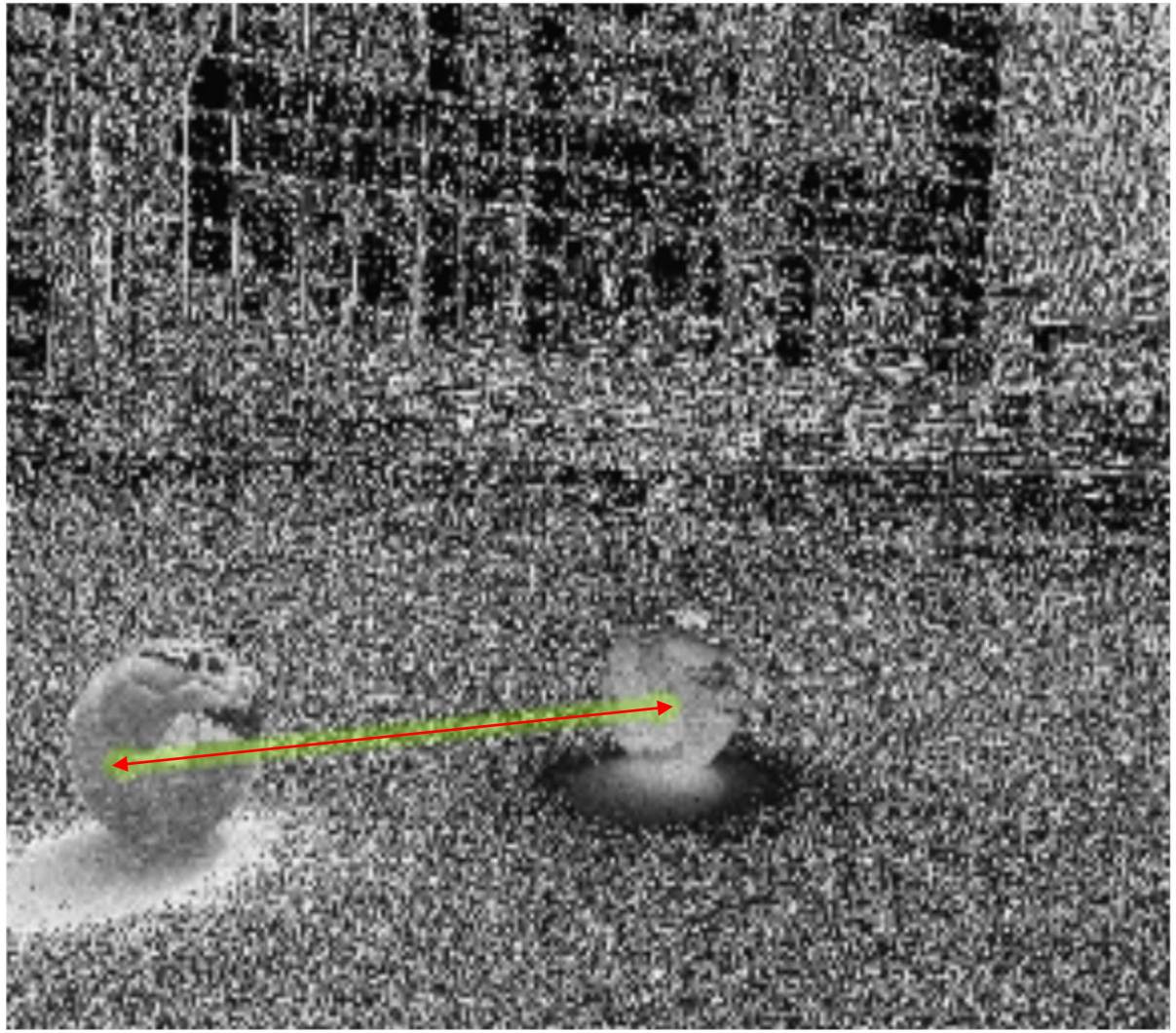
- Acquisition et conversion en niveaux de gris des images pour pouvoir effectuer une détection de contours
- Programmation de la transformée de Hough pour la détection de la balle
- Déterminer les conditions nécessaires et suffisantes pour dire que la balle a franchit la ligne de but

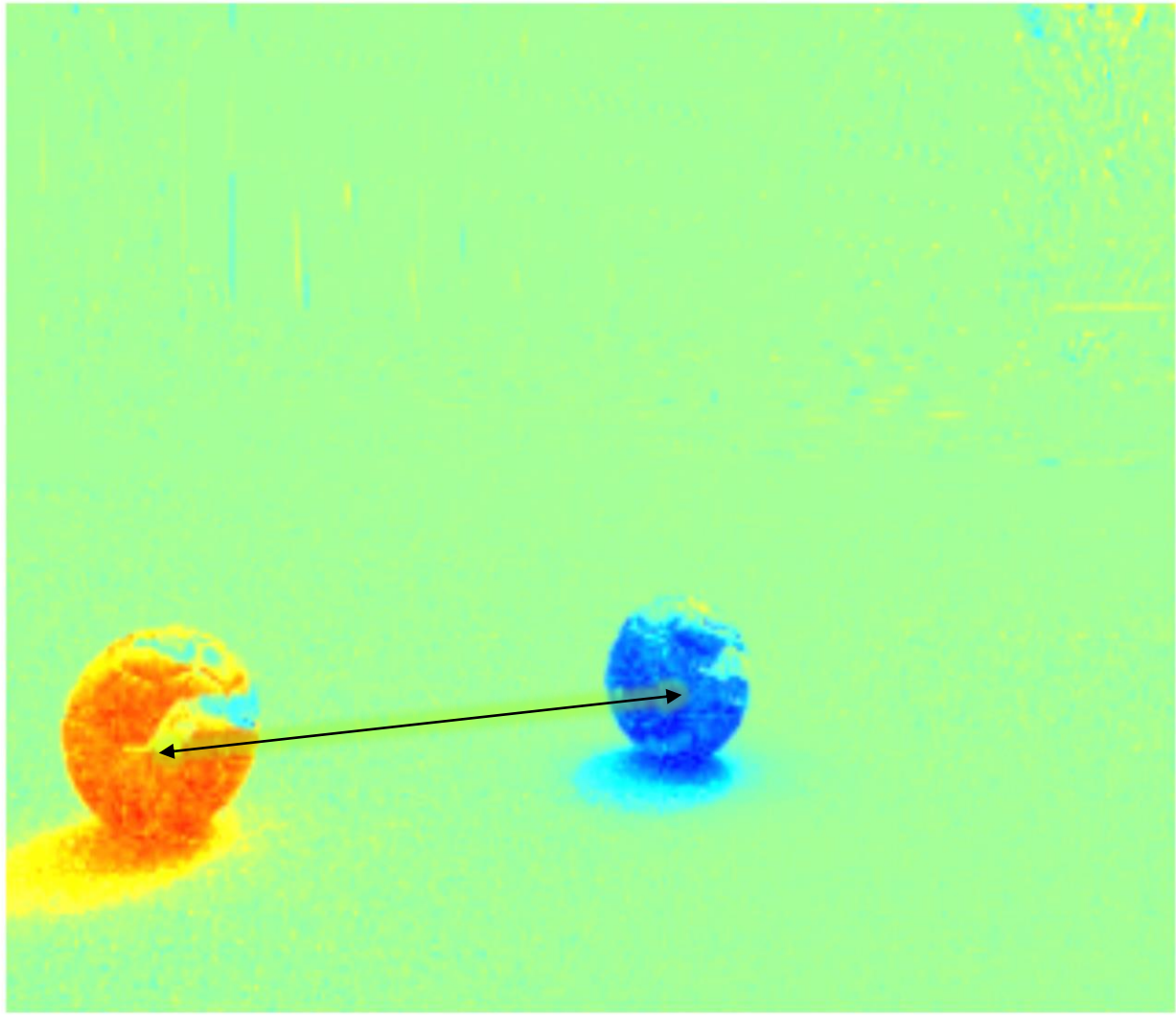
LA DEMARCHE EN RESUME

- 1-Acquisition d'images et segmentation d'images
- 2- Détection de contours par utilisation de la Transformée de Hough
- 3- Reconnaisse de balle
- C- Suivi de balle

B-a) Observation du déplacement de la balle

SOUSTRACTION D'IMAGES





B-b) DETERMINATION DES CONTOURS PAR PLUSIEURS METHODES

- 1- Méthode qui consiste à ne pas utiliser le gradient (ANNEXE 2)
- 2- Méthodes utilisant le gradient (ANNEXE 1)

  a-) filtre dérivateur

  b-) filtre de Kirsch
- 3-Comparaison des filtres

- $A: \mathbb{R}^2 \rightarrow \mathbb{R}^2$,
- $\frac{\partial A}{\partial x}\overrightarrow{U_x} + \frac{\partial A}{\partial y}\overrightarrow{U_y} = \vec{\nabla} A$
- $\boldsymbol{\partial A/\partial x = (A(x+\mathbb{d}x, y) - A(x,y))/\mathbb{d}x}$
- Mais l'image a un caractère discret donc :
- $A(x, y) \Leftrightarrow a_i, l$

COMPARAISON DE DIFFERENTS FILTRES

- On choisira finalement des filtres données par les modules python pour un gain de temps

FILTRE DERIVATEUR

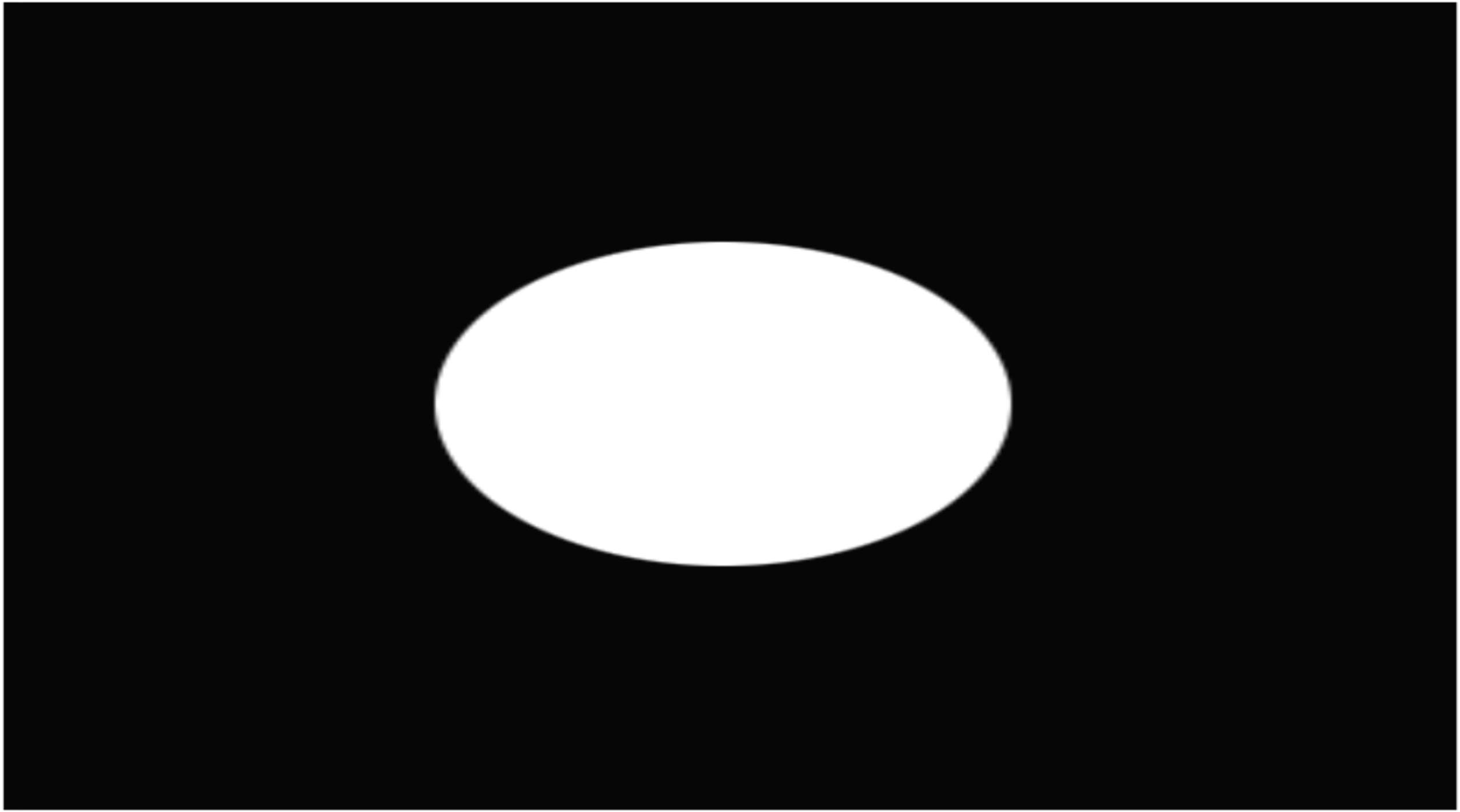
FILTRE (ECART-TYPE)

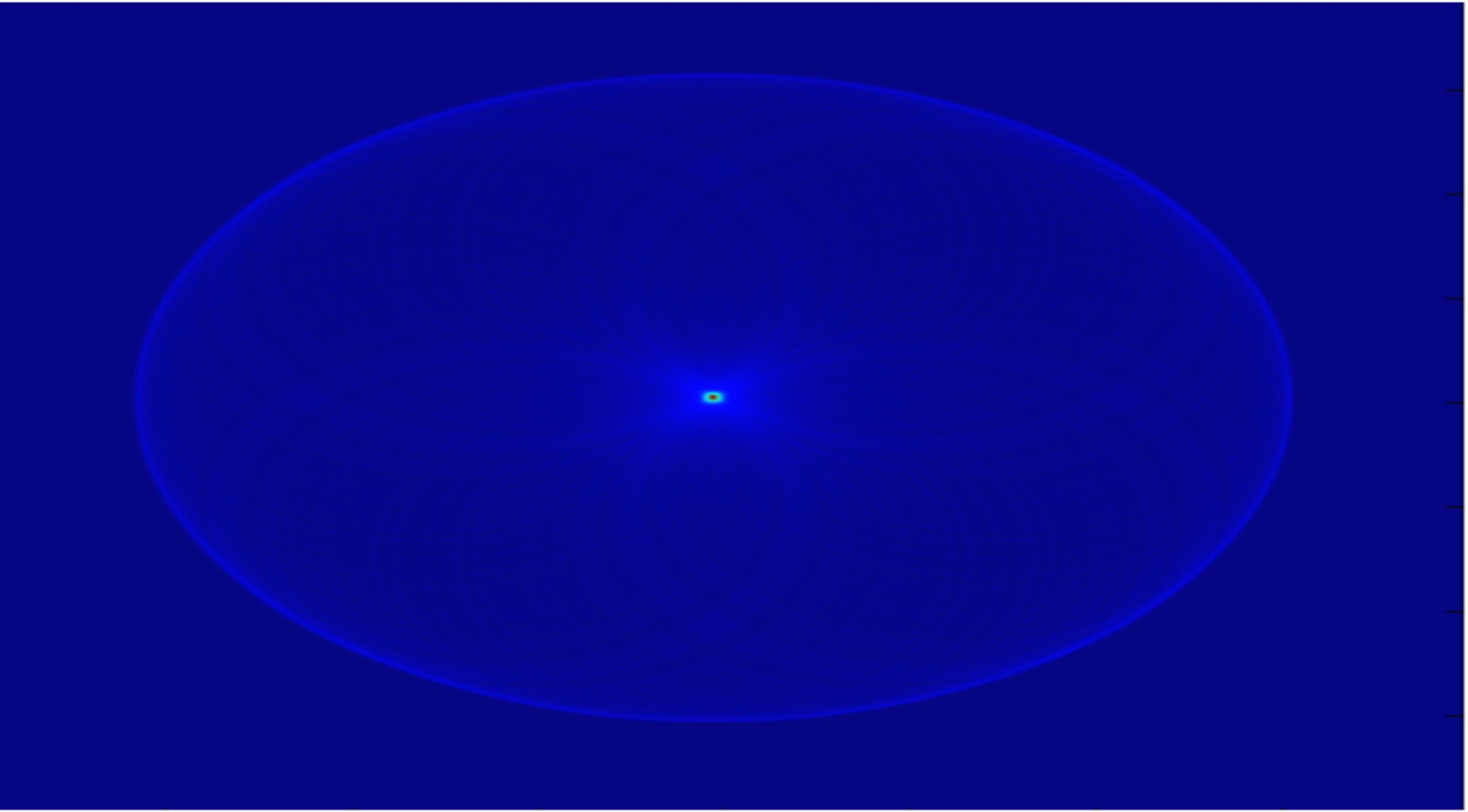
FILTRE DE KIRSH

FILTRE(python)

B-c) TRANSFORMEE DE HOUGH

- Cadre d'application
- -Détections de cercles
  - $(x-\alpha)^2+(y-\beta)^2=R^2 \qquad (\alpha, B) \in \mathbb{R}^2$
- Comment il s'applique t'il ?
  - $\alpha = x - R\cos(\theta)$
  - $\beta = y - R\sin(\theta)$
- Applications (ANNEXE 3)
- Resultats





SUIVI DE BALLE

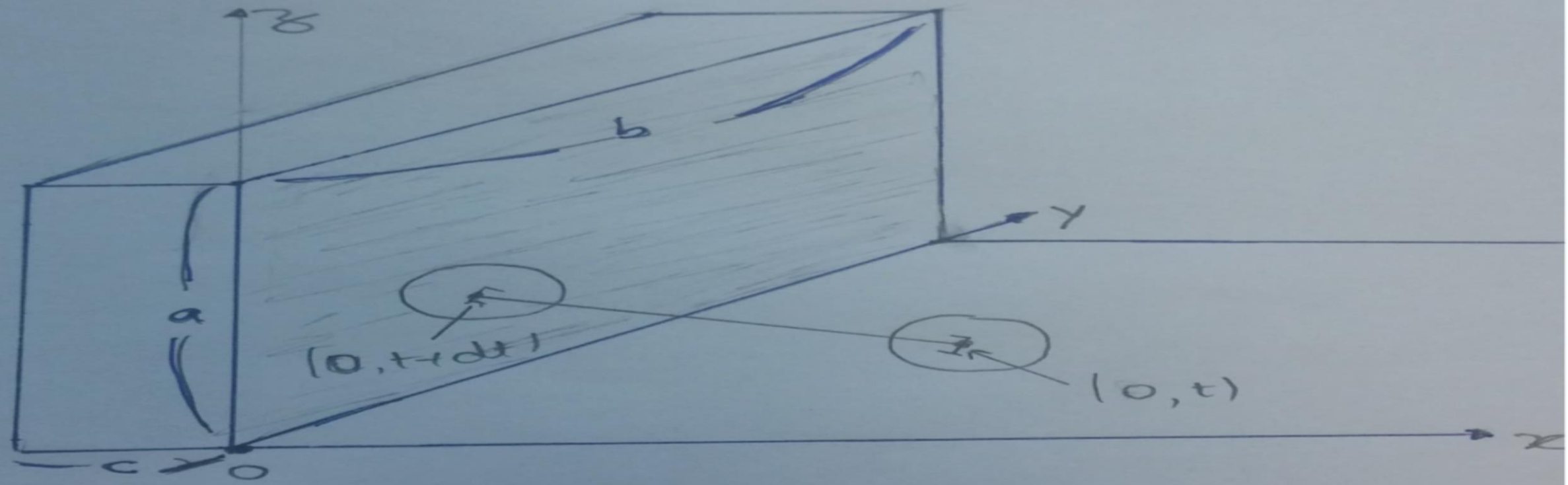
- On se limitera au plan (O,x,y) pour réduire la complexité du problème
- Avec la transformée de hough on a su localiser le centre d'une balle

On peut donc suivre la balle sur le terrain à tout moment

T4
T6
T1
T2
T3
T5

CONCLUSION

- Temps mis et complexité des algorithmes de détections de contours par méthode de gradients
- Temps et complexité mis par la transformée de hough
- Suivi de Balle
- CNS pour considérer qu'il y'a but


z
b
y
a
(0, t+dt)
(0, t)
x
c
O

ANNEXES:

- Filtres utilisant le gradient
- Filtre basé sur l'écart-type
- Transformée de Hough
- Conversion en niveaux de gris et seuillage

```
A = mpimg.imread('/Users/tewfik/Desktop/ballonR3.png')

R = (A[:,:,0])

G = (A[:,:,1])

B = (A[:,:,2])

A = np.floor(255*(0.2125*R + 0.7154*G + 0.0721*B))

plt.axis('off')
plt.figure(1)
plt.imshow(A,cmap=cm.gray)
plt.show()

def Convolution(A,Conv):
    n,m = A.shape
    ic = np.shape(Conv)[0]//2
    B = np.zeros((n,m))
    for i in range(ic,n-ic):
        for j in range(ic,m-ic):
            for ik in range(-ic,ic+1):
                for jk in range(-ic,ic+1):
                    B[i,j] +=A[i+ik][j+jk]*Conv[ik+ic][jk+ic]
    return B

GradX=np.array([[-1,-1,-1],[0,0,0],[1,1,1]])
GradY=np.array([[-1,0,1],[-1,0,1],[-1,0,1]])

DetContour = np.sqrt(Convolution(A,GradX)**2+Convolution(A,GradY)**2)

Aseuil = np.array(DetContour)>7
Aseuilneg = np.max(Aseuil) - Aseuil
plt.figure(2)
plt.imshow(Aseuilneg,cmap =cm.gray)
plt.show()
```

```
 6
 7  A = mpimg.imread('/Users/tewfik/Desktop/ballonR3.png')
 8  taille = A.shape[0]*A.shape[1]
 9
10  R = (A[:,:,0])
11
12  G = (A[:,:,1])
13
14  B = (A[:,:,2])
15
16  A = np.floor(255*(0.2125*R + 0.7154*G + 0.0721*B))
17
18  plt.axis('off')
19  plt.figure(1)
20  plt.imshow(A,cmap=cm.gray)
21  #plt.show()
22
23
24  def filtre_std(A,taille):
25      n,m =A.shape
26
27      ic = taille//2
28      B=np.zeros((n,m))
29      for i in range(ic,n-ic):
30          for j in range(ic,m-ic):
31              liste = []
32              for ik in range (-ic,ic+1):
33                  for jk in range(-ic,ic+1):
34                      liste.append(A[i+ik,j+jk])
35              B[i,j] = np.std(liste)
36      return B
37
38
39  Af = filtre_std(A,3)
40  Aseuil = np.array(Af)>5
41  Aseuilneg = np.max(Aseuil) - Aseuil
42  plt.figure(2)
43  plt.imshow(Aseuilneg,cmap =cm.gray)
44  plt.show()
```

```
def houghtransform(D):
    m = D.shape[0]
    n = D.shape[1]
    p = np.sqrt((n)**2+ m**2)
    Matrice = np.zeros((m,n))
    # print 'partie entiere de p: ',int(p)
    # print Matrice.shape
    for i in range(m):
        for j in range(n):
            if D[i,j]>0:
                for theta in range(0,360):
                    a= i-int(328.20*np.cos(theta*np.pi/180))
                    b= j-int(328.20*np.sin(theta*np.pi/180))
                    #print (a,b,R)
                    if a<m and b<n:
                        Matrice[a,b]+=1
    return Matrice

#Autre possiblité
# def houghtransform2(D):
#     m = D.shape[0]
#     n = D.shape[1]
#     p = n/5
#     Matrice = np.zeros((p,m,n),dtype=np.int8)
#     for i in xrange(m):
#         for j in xrange(n):
#             if D[i,j]>0:
#                 for x0 in range(m):
#                     for y0 in range(n):
#                         R = int(np.sqrt(((i-x0)**2)+((j-y0)**2)))
#                         if R<(n/5):
#                             Matrice[R,x0,y0]+=1
#     return Matrice


Accumulation = houghtransform(Ac)

fig,ax = plt.subplots(1)
ax.set_aspect('equal')
# Show the image
ax.imshow(A,cmap=cm.gray)

circ = Circle((60, 60), 55)
ax.add_patch(circ)
# A = img.imread(filename)
```

```
C = img.imread('/Users/tewfik/Desktop/m2.png')

D = img.imread('/Users/tewfik/Desktop/m24.png')

def luminosité(tripletrgb): ## renvoie la valeur entière de la luminosité
    r,g,b = tripletrgb
    return int(0.2125*r+0.7154*g+0.0721*b)

def niveaudegris(Imagergb): ## renvoie l'image en niveaux de gris en utilisant la fonction luminosité
    a= Imagergb.shape[0]
    b =Imagergb.shape[1]

    im = IMG.new('L',(a,b))
    pix = im.load()
    pixrgb = Imagergb[a,b,0]
    for x in range(a):
        for y in range(b):
            pix[x,y] = luminosité(pixrgb[x,y])
    return im

A = niveaudegris(C)
plt.imshow(A)
plt.show()


imgris= niveaudegris(C)

def seuillage(pixel,seuil):
    if pixel< seuil:
        return 0
    return 255

c,d = imgris.size
imnb = IMG.new('L',(c,d))
pixgris = imgris.load()
pixnb = imnb.load()
for i in range(c):
    for j in range(d):
        pixnb[i,j] = seuillage(pixgris[i,j],100)

imnb.show()
```