REBOND D'UNE BALLE DE TENNIS : AIDE À L'ARBITRAGE HUMAIN

INTRODUCTION

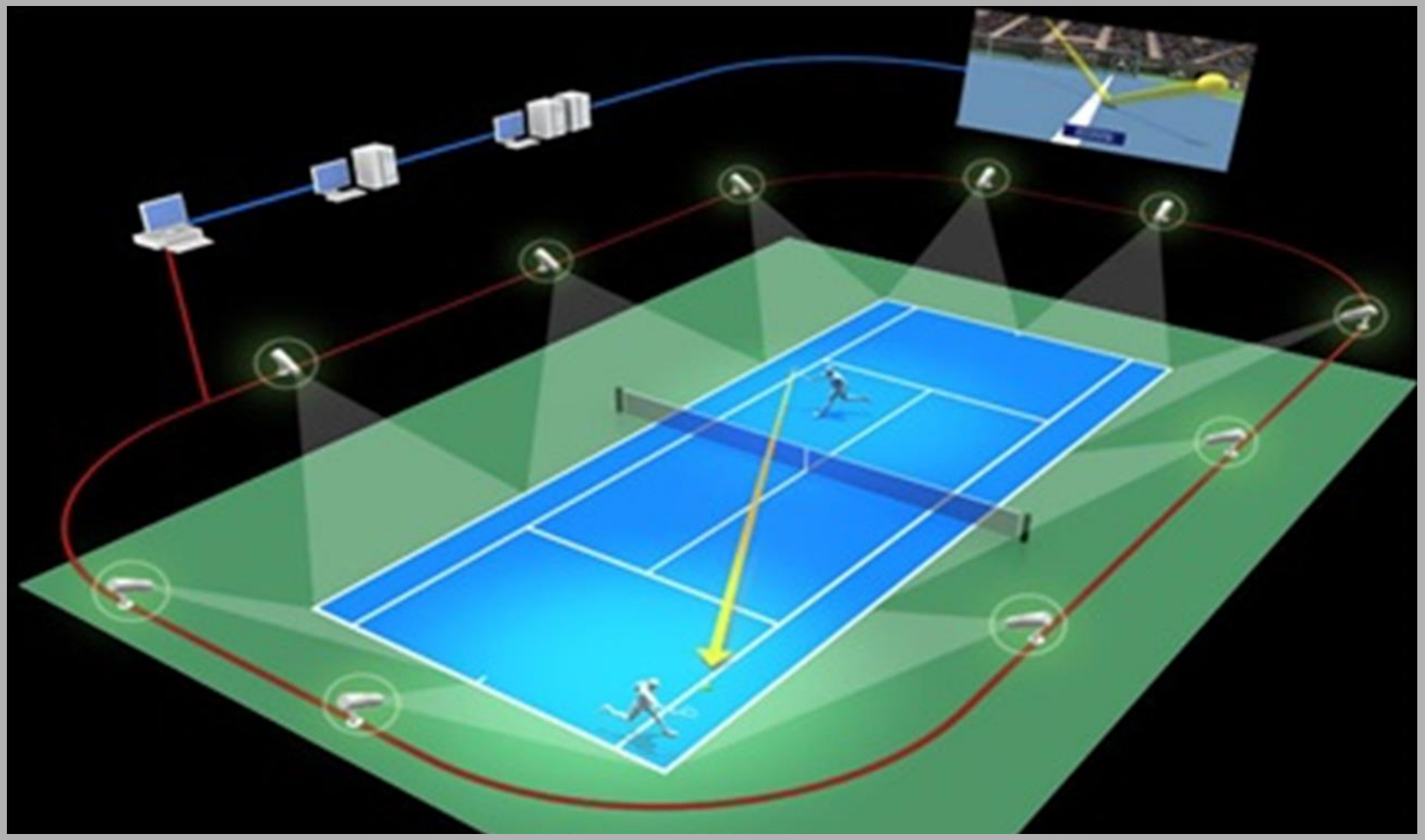
PROBLÉMATIQUE

Comment aider l'arbitrage humain par différents moyens technologiques ?

PLAN

1. Etude du système Hawk-eye
2. Etude d'une alternative
3. Conclusion

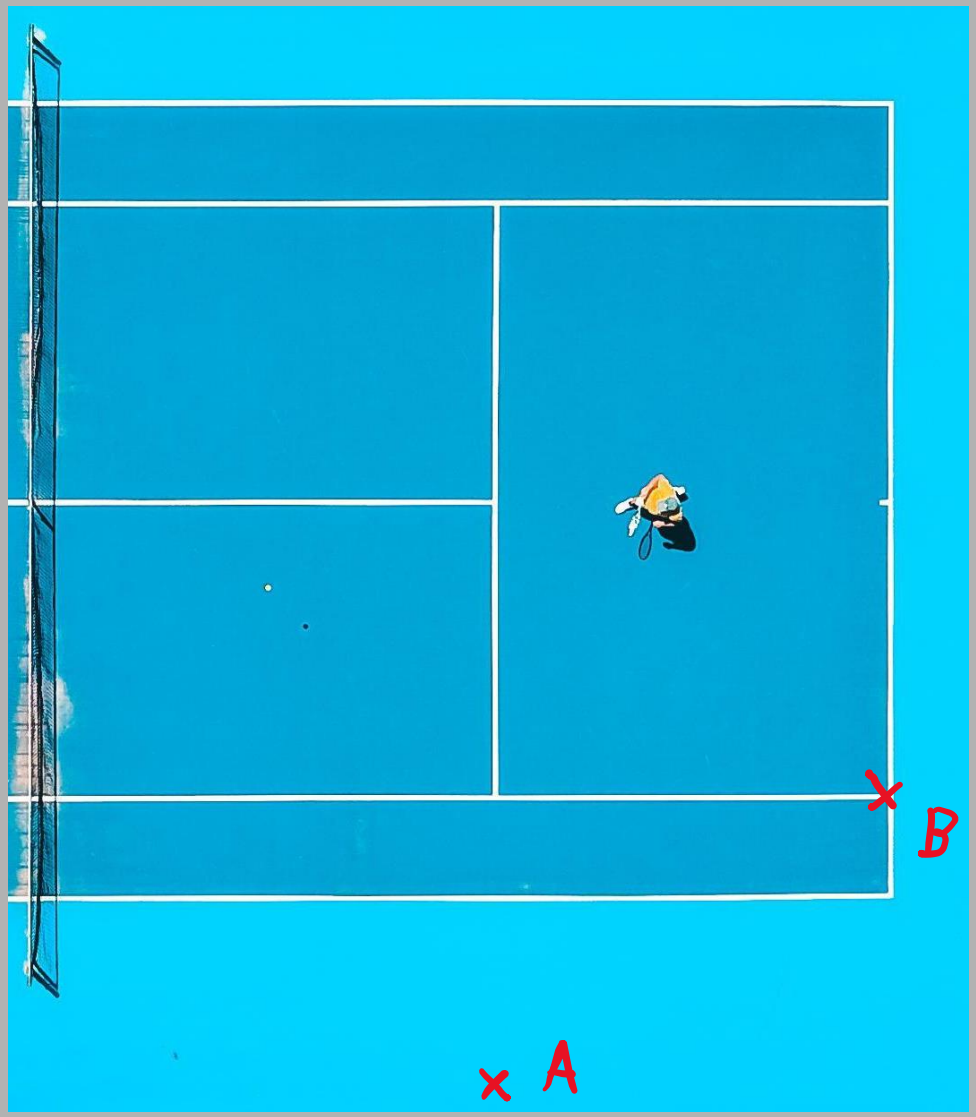
FONCTIONNEMENT



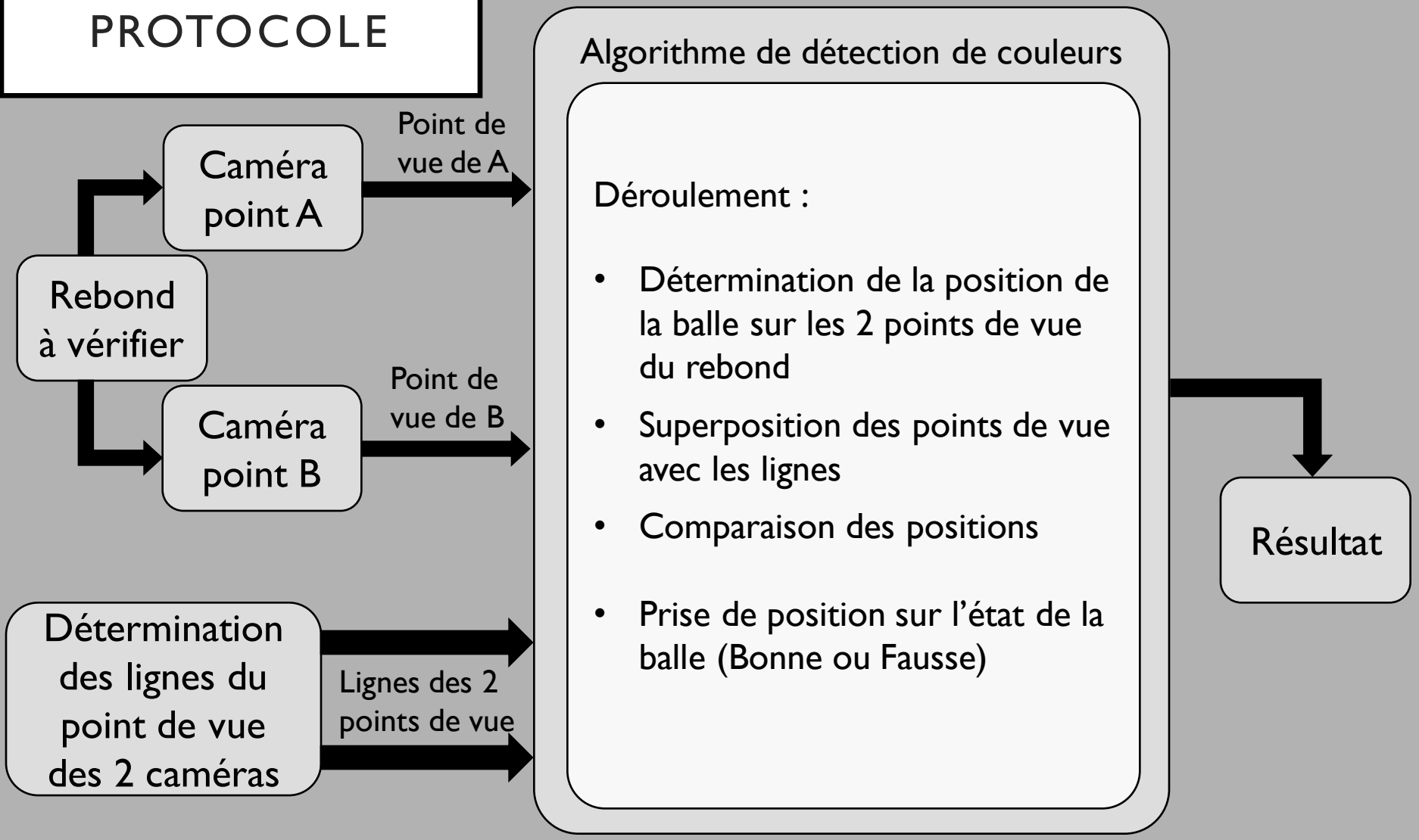
MODÉLISATION

Hypothèses :

- Caméra horizontale centrée sur le rebond
- Balle considérée comme un pixel assimilé à son centre

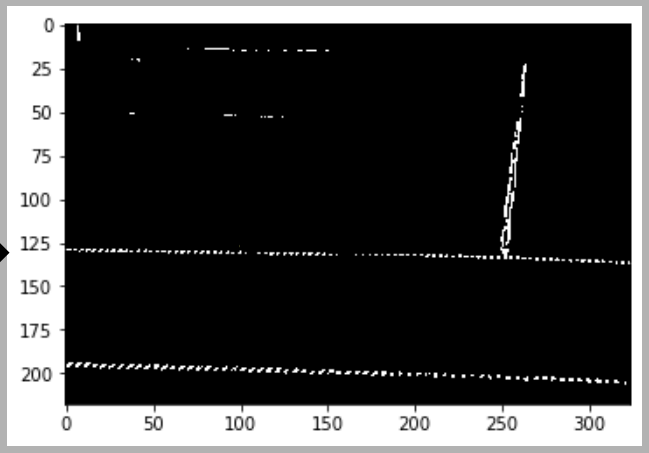


PROTOCOLE


Caméra point A
Point de vue de A
Rebond à vérifier
Caméra point B
Point de vue de B
Détermination des lignes du point de vue des 2 caméras
Lignes des 2 points de vue
Algorithme de détection de couleurs
Déroulement :
• Détermination de la position de la balle sur les 2 points de vue du rebond
• Superposition des points de vue avec les lignes
• Comparaison des positions
• Prise de position sur l'état de la balle (Bonne ou Fausse)
Résultat

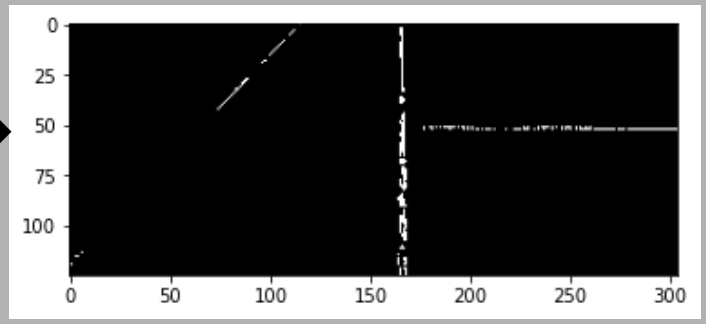
TESTS ET RÉSULTATS

Point de vue de A



Détermination des lignes du point de vue des 2 caméras

Point de vue de B



TESTS ET RÉSULTATS

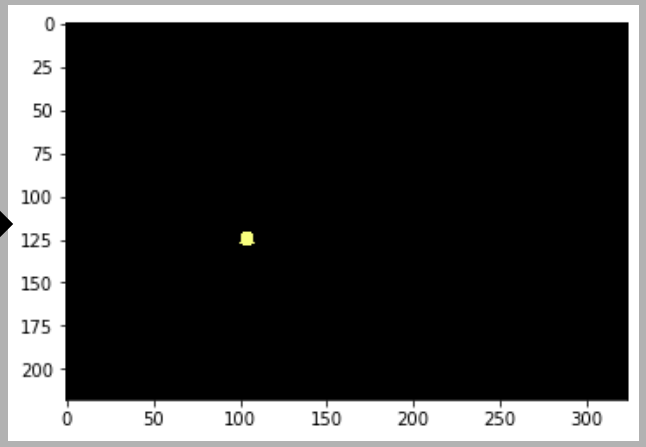
Image d'un service à ≈ 80km/h

Algorithme de détection de couleurs

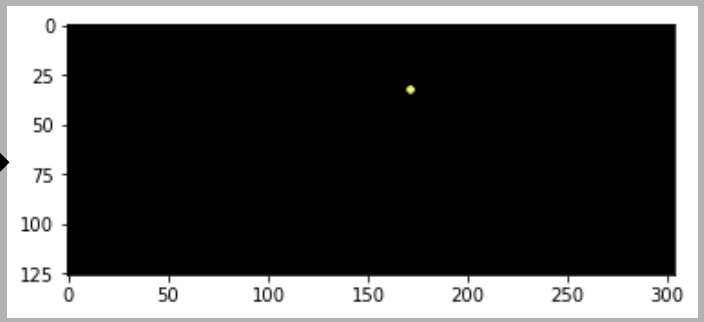
Déroulement :

- Détermination de la position de la balle sur les 2 points de vue du rebond
- Superposition des points de vue avec les lignes
- Comparaison des positions
- Prise de position sur l'état de la balle (Bonne ou Fausse)

Point de vue de A



Point de vue de B



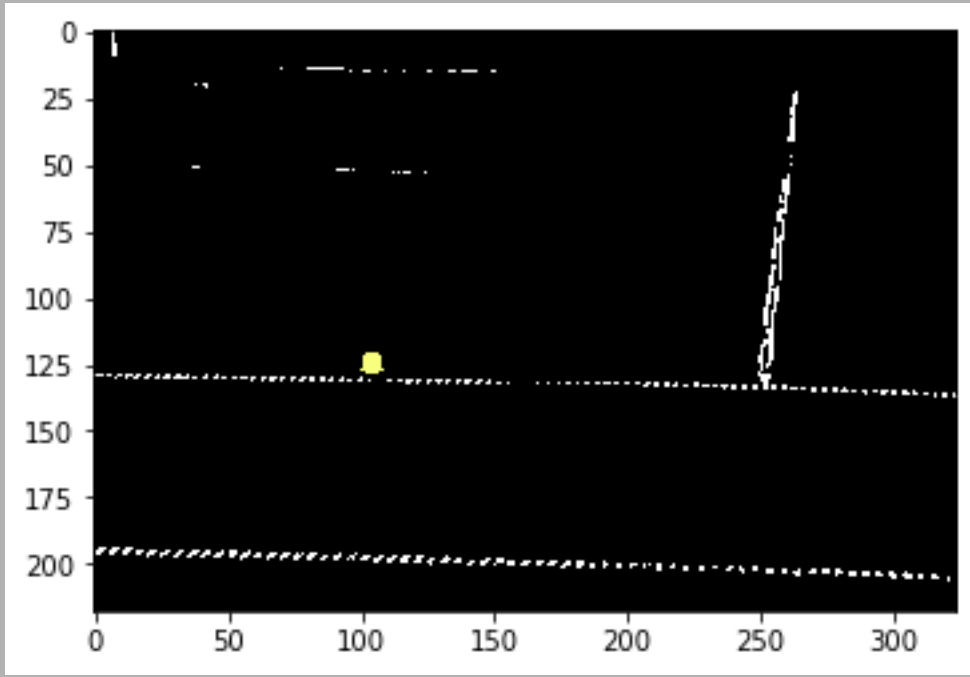
TESTS ET RÉSULTATS

Algorithme de détection de couleurs

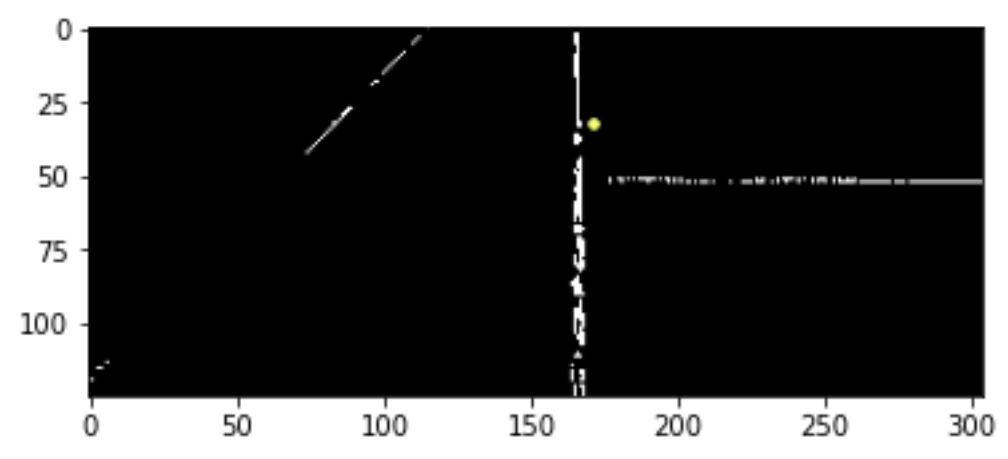
Déroulement :

- Détermination de la position de la balle sur les 2 points de vue du rebond
- Superposition des points de vue avec les lignes
- Comparaison des positions
- Prise de position sur l'état de la balle (Bonne ou Fausse)

Point de vue de A

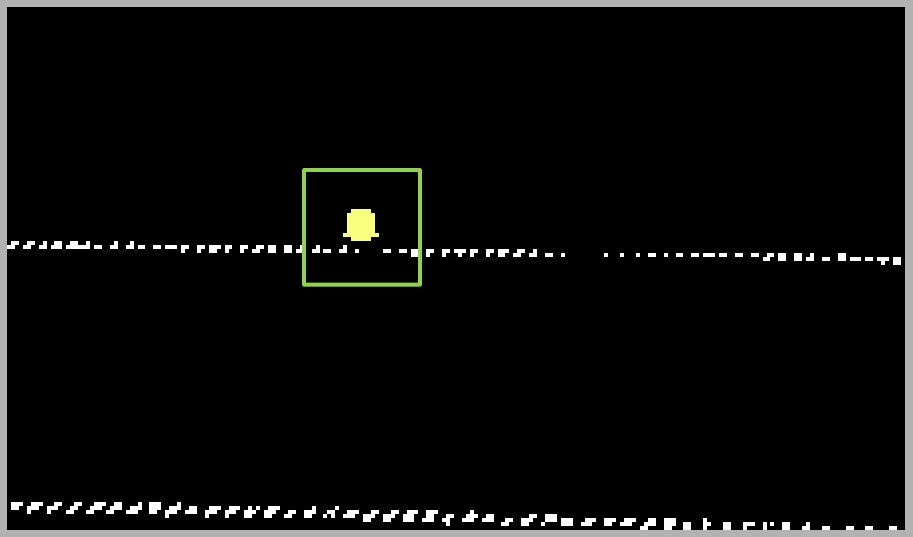


Point de vue de B



TESTS ET RÉSULTATS

Point de vue de A

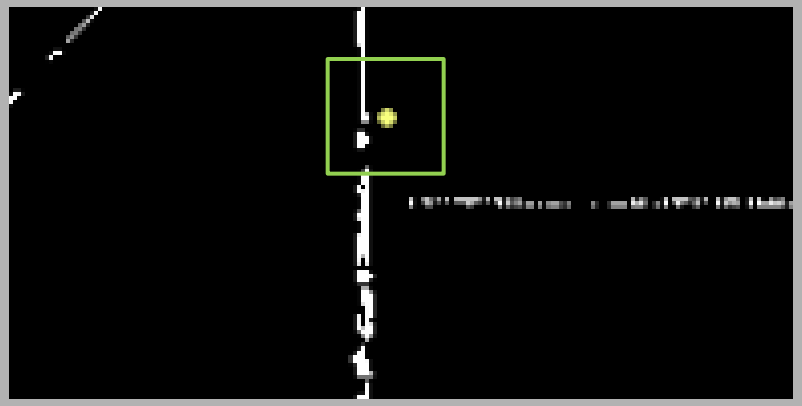


Algorithme de détection de couleurs

Déroulement :

- Détermination de la position de la balle sur les 2 points de vue du rebond
- Superposition des points de vue avec les lignes
- Comparaison des positions
- Prise de position sur l'état de la balle (Bonne ou Fausse)

Point de vue de B



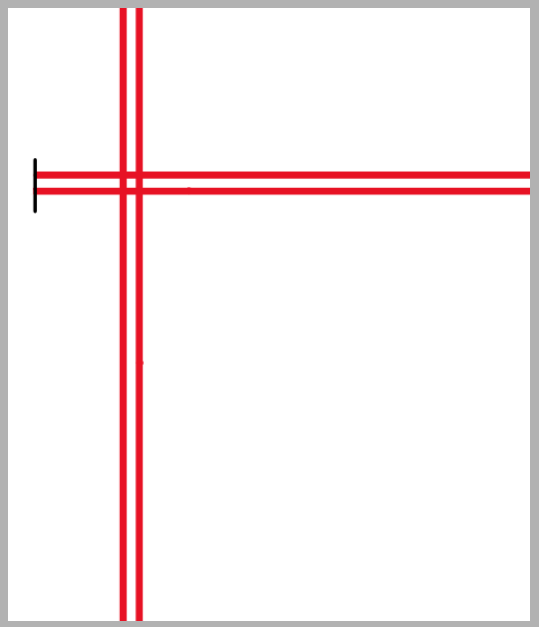
TESTS ET RÉSULTATS

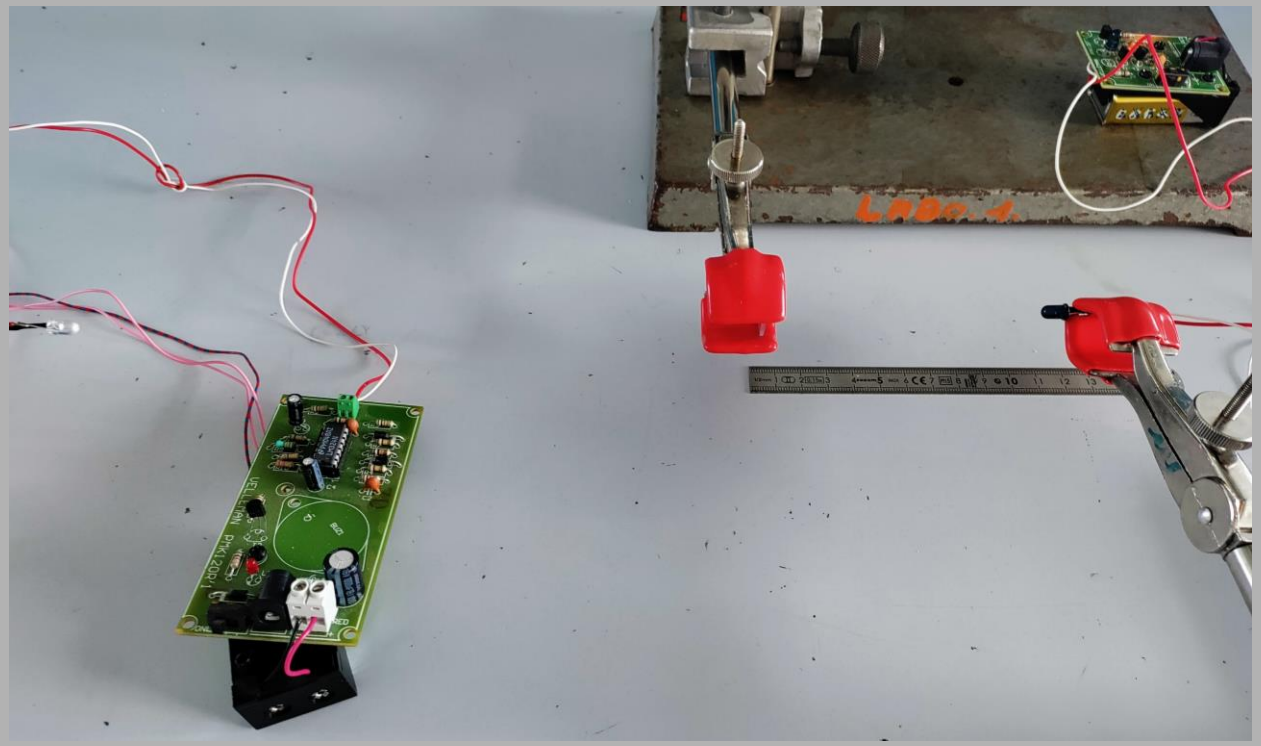
vitesse en km/h	30		50		70		90		110		130	
état de la balle déterminée par l'homme	4	4	4	4	4	4	4	4	4	4	4	4
état de la balle déterminée par l'algorithme	4	4	4	4	4	4	3	5	2	5	4	2
rebonds indéterminables par la vidéo	0		0		0		0		1		2	

PROBLÈMES NOTABLES

- Imprécisions si pas assez d'images par secondes ou si balle trop rapide

IDÉE ET MISE EN PLACE





TESTS ET LIMITES

Temps de rebond d'une balle :
entre 1,8 et 20 ms

CONCLUSION

Persistance de l'oeil et effet phi du cerveau

- modélisation de l'oeil par une caméra à une vingtaine d'images par seconde

	œil simple	**Hawk-eye**	**alternative**
fréquence d'échantillonnage	20 ips	Jusqu'à 1000 ips	20kHz
écart de position entre 2 « images » à 250km/h	3,47 m/image	6,94 cm/image	3,47 cm/image
prix	Bénévole dans les grands tournois	70 000 € la semaine	15€

Merci pour votre attention !

```python
import matplotlib.pyplot as plt
import matplotlib.image as img
import numpy as np
import copy as cp

image_A_balle = img.imread('point A rebond 6.jpg').tolist()
image_B_balle = img.imread('point B rebond 6.jpg').tolist()

image_A_ligne = img.imread('point A.jpg').tolist()
image_B_ligne = img.imread('point B.jpg').tolist()

couleur_centre = [251,255,155]
couleur_bande = [255,183,155]


def trouver_pixels_de_couleur(image, couleur):
    pixels_trouves = []
    largeur, hauteur = len(image[0]),len(image)

    if couleur == couleur_centre :
        for x in range(hauteur):
            for y in range(largeur):
                pixel = image[x][y]
                if pixel == couleur:
                    pixels_trouves.append((x, y))

    elif couleur == couleur_bande :
        for x in range(hauteur):
            for y in range(largeur):
                pixel = image[x][y]
                if pixel[0] >= couleur_bande[0] and pixel[1] >= couleur_bande[1] and pixel[2] >= couleur_bande[2]:
                        pixels_trouves.append((x, y))

    return pixels_trouves
```

```python
def image_bande():
    bande_A = np.zeros_like(image_A_ligne).tolist()
    bande_B = np.zeros_like(image_B_ligne).tolist()
    pixel_A = trouver_pixels_de_couleur(image_A_ligne, couleur_bande)
    pixel_B = trouver_pixels_de_couleur(image_B_ligne, couleur_bande)

    for i in pixel_A:
        bande_A[i[0]][i[1]]=image_A_ligne[i[0]][i[1]]
    for i in pixel_B:
        bande_B[i[0]][i[1]]=image_B_ligne[i[0]][i[1]]

    bandes = (bande_A,bande_B)

    for a in range(2):
       for i in range(len(bandes[a])):
           for j in range(len(bandes[a][0])):
               if bandes[a][i][j]!=[0,0,0]:
                   bandes[a][i][j]=[255,255,255]

    return bandes

images = image_bande()
```

```python
def pixels_interessants(images):
    pixels = []

    # rebond A
    for i in range(120,140):
        for j in range(len(images[0][0])):
            if images[0][i][j]!=[0,0,0]:
                pixels.append((i,j))

    # rebond B
    for i in range(40,60):
        for j in range(len(images[1][0])):
            if images[1][i][j]!=[0,0,0]:
                pixels.append((i,j))

    return pixels
```

```python
def image_balle():
    balle_A = np.zeros_like(image_A_balle)
    balle_B = np.zeros_like(image_B_balle)
    pixel_A = trouver_pixels_de_couleur(image_A_balle, couleur_centre)
    pixel_B = trouver_pixels_de_couleur(image_B_ligne, couleur_centre)

    for i in pixel_A:
        balle_A[i[0]][i[1]]=image_A_balle[i[0]][i[1]]
    for i in pixel_B:
        balle_B[i[0]][i[1]]=image_B_balle[i[0]][i[1]]

    return (balle_A,balle_B)

balles = image_balle()
```

```python
def superposition(balles,images):
    rebonds = (cp.deepcopy(images[0]),cp.deepcopy(images[1]))

    for a in range(2):
        for i in range(len(balles[a])):
            for j in range(len(balles[a][0])):
                if balles[a][i][j]!=[0,0,0]:
                    rebonds[a][i][j]=balles[a][i][j]

    return rebonds
```

```python
def verif_position(balles,images):
    delta = 15
    pixels_bandes = pixels_interessants(images)
    pixels_balle = []
    for a in range(2):
        for i in range(len(balles[a])):
            for j in range(len(balles[a][0])):
                if balles[a][i][j]!=[0,0,0]:
                    pixels_balle.append((i,j))

    k=0
    for a in range(2):
        mid_l = pixels_balle[a][1]
        mid_h = pixels_balle[a][0]
        mean = [0,0]
        count = 0
        for i in range(mid_h-delta,mid_h+delta):
            for j in range(mid_l-delta,mid_l+delta):
                if (i,j) in pixels_bandes:
                    mean[0]+=i
                    mean[1]+=j
                    count+=1
        pixel_m = (mean[0]/count,mean[1]/count)

        if a == 0:
            if pixels_balle[0][0]>=pixel_m[0] and pixels_balle[0][1]<=pixel_m[1]:
                k+=1
        if a == 1:
            if pixels_balle[0][0]>=pixel_m[0] and pixels_balle[0][1]>=pixel_m[1]:
                k+=1
    if k==2:
        return "La balle est bonne !"
    else :
        return "FAUTE !!"
```

- https://www.01net.com/tests/gopro-hero-lcd-fiche-technique-25959.html
- The Serve Impact in Tennis: First Large-Scale Study of Big HawkEye Data : https://sci-hub.live/10.1002/sam.11316
- https://www.europe1.fr/emissions/L-innovation-du-jour/tennis-un-boitier-qui-indique-si-la-balle-est-in-ou-out-3901948
- HAWK-EYE TENNIS SYSTEM : https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1341323&isnumber=29549
- CONCOURS COMMUNS POLYTECHNIQUES : Système d'aide à l'arbitrage : Le HAWKEYE : PSI Informatique 2018
- https://www.lunion.fr/id490811/article/2023-06-04/ca-fait-debat-la-redac-est-il-pertinent-de-remplacer-les-juges-de-ligne-de#:~:text=En%202025%2C%20une%20nouvelle%20technologie,des%20balles%20sur%20le%20court
- https://www.atptour.com/en/news/electronic-line-calling-release-april-2023
- https://www.youtube.com/watch?v=fachbvEuTak
- https://technis.com/fr/technis-capture/
- https://hitek.fr/actualite/frame-rate-jeux-videos-oeil_6959