Reconnaissance Faciale par Infrarouge

54053
Nello Barut

Reconnaissance Faciale : Généralités

Bénéfices Utilisateur

- Rapidité
- Fiabilité
- Ergonomie

Santé / Prévention

Assurer la sécurité
des utilisateurs d'une piscine

Plan

1. Obtention d’un spectre
2. Traitement du spectre
3. Méthodes de comparaison
4. Application pratique
5. Limites et optimisations
6. Annexes

Comment créer une méthode
de reconnaissance faciale
permettant d'identifier des individus
pour suivre leurs déplacements
dans un espace clos ?

Obtention d'un spectre à partir des fichiers d'une caméra thermique

Mise en place

Dispositif de prise de vue

Acquisition

Obtention d'un spectre à partir des fichiers d'une caméra thermique

la caméra thermique

Modèle : FLIR E 60

Caractéristiques

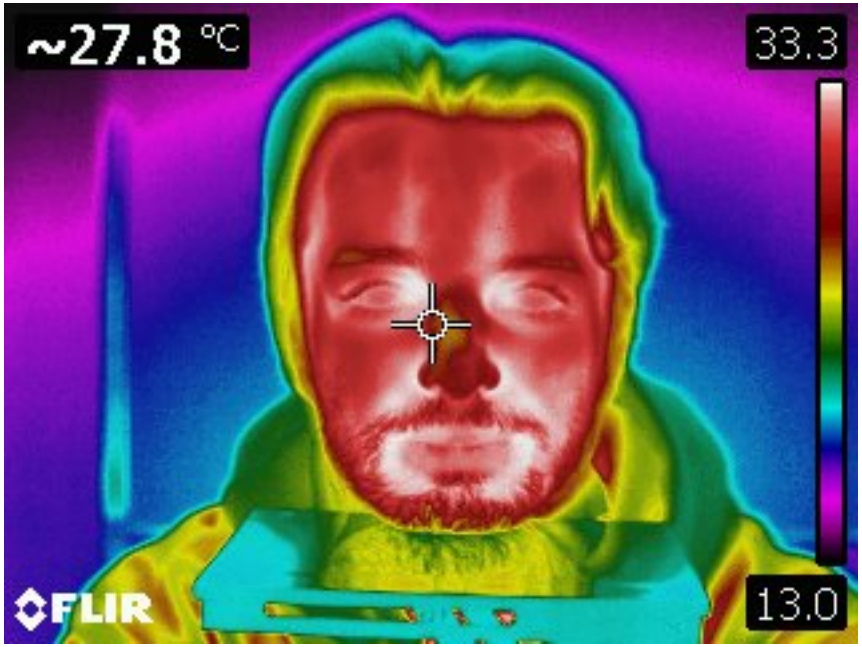
- Résolution IR : 320 x 240 pixels
- Plage de mesure des températures : -20 à 650°C
- Taux de rafraichissement : 60Hz
- Sensibilité thermique : < 0,05°C

Fonctionnement

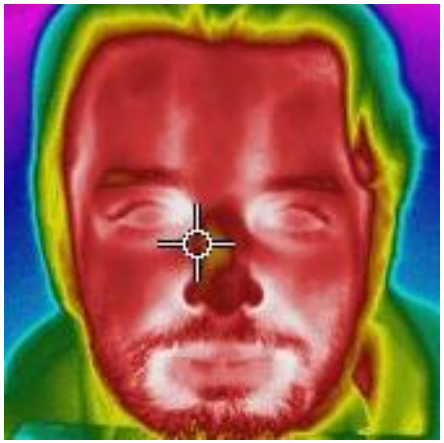
- Capteur de flux thermique
- Loi de Stefan
  pour un corps gris d'émissivité ε :
  $\Phi = \varepsilon \sigma T^4$

Obtention d'un spectre à partir des fichiers d'une caméra thermique

l'image



Fichier original



Fichier recadré

Obtention d'un spectre à partir des fichiers d'une caméra thermique

les fichiers CSV

```python
def CSVcrop(T,Image):
    Tcrop=[]

    yh,yb,decalage=Taille(Image)
    box=(160-decalage-(yb-yh)//2,yh,160-decalage+(yb-yh)//2,yb)

    for i in range(box[1],box[3]):
        TcropTempo=[]
        for k in range(box[0],box[2]):
            TcropTempo.append(T[i][k])
        Tcrop.append(TcropTempo)
    return(Tcrop)
```

Mise en correspondance du fichier de données

```python
def Moyenne(Matrice):
    somme=0
    for i in range(0,len(Matrice)):
        for k in range(0,len(Matrice[i])):
            somme=somme+Matrice[i][k]

    moy=somme/(len(Matrice)*len(Matrice[0]))

    for i in range(0,len(Matrice)):
        for k in range(0,len(Matrice[i])):
            Matrice[i][k]=Matrice[i][k]-moy

```

Etalonnage du fichier des températures

Obtention d'un spectre à partir des fichiers d'une caméra thermique

la transformée de Fourier

$$F(\sigma_x, \sigma_y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x,y) e^{-i2\pi(\sigma_x x + \sigma_y y)} dx dy$$

Transformée de Fourier à 2 variables

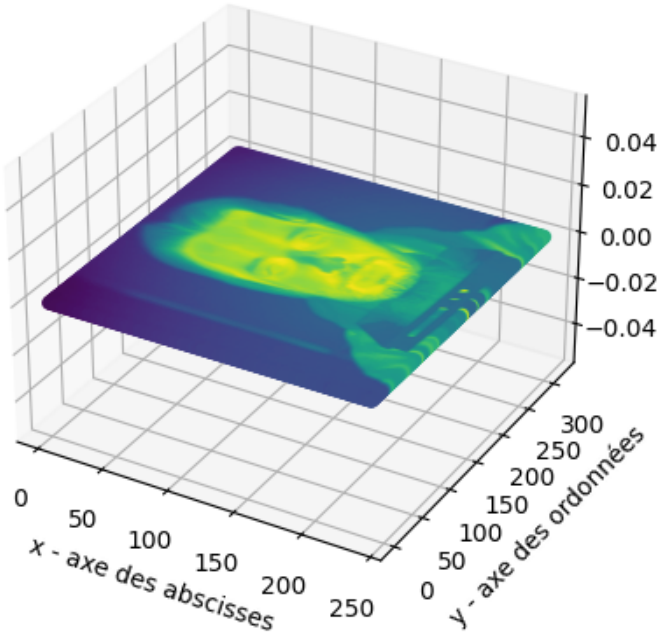
$$S(\sigma_x, \sigma_y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) e^{-i2\pi(\frac{\sigma_x m}{M} + \frac{\sigma_y n}{N})}$$

Transformée de Fourier discrétisée (FTD)

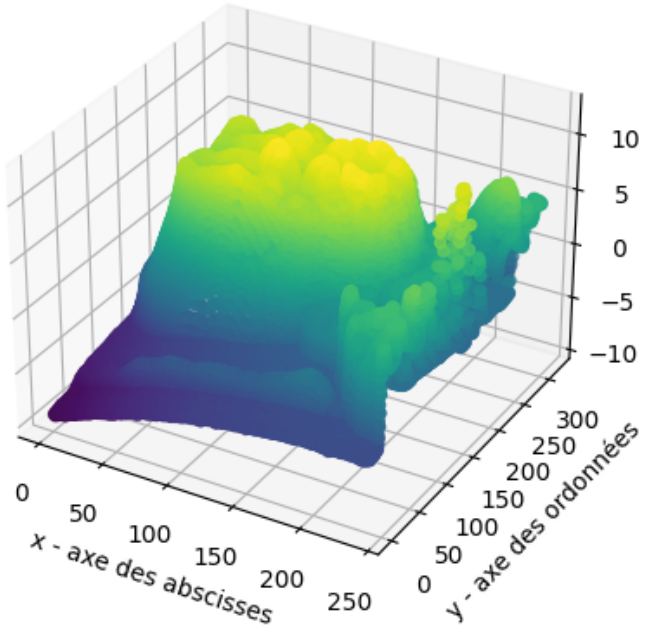
FFT : Algorithme de résolution de la Transformée de Fourier

Obtention d'un spectre à partir des fichiers d'une caméra thermique

Représentations Graphiques



Représentation 2D des températures

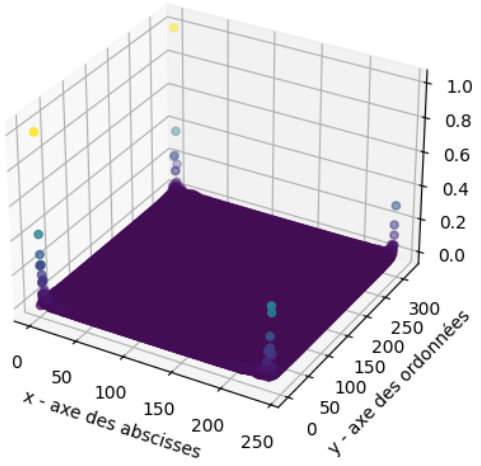


Représentation 3D des températures

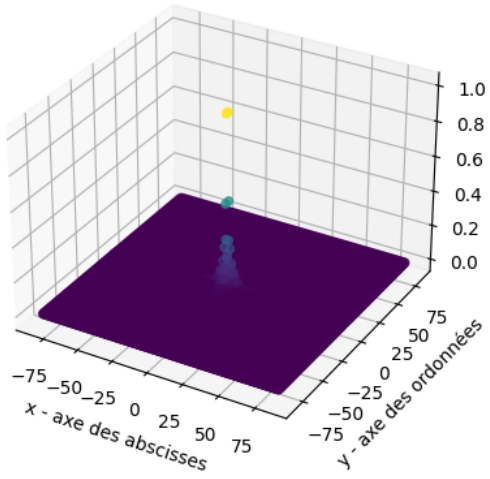
Traitements des spectres

réduction des spectres :

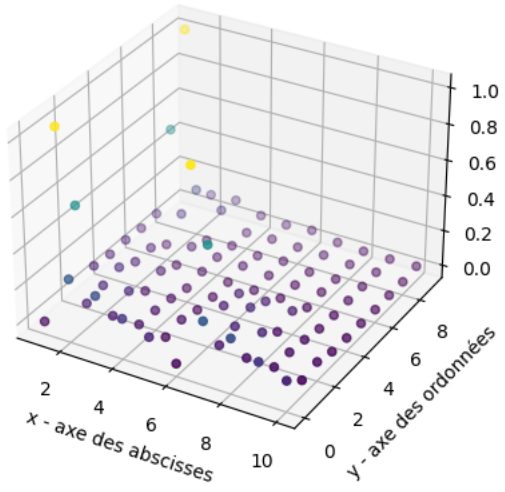
conservation des valeurs critiques



Spectre FFT Initial



Spectre FFT



Spectre FFT Réduit

Utilisation des spectres

Base de données

Caractéristiques

- 15 individus
- 5 photos et fichiers liées par individus
- 4 photos « données » + 1 photo « test »

Méthode

- Moyenne et / ou intervalles de valeurs sur les 4 photos « données »

Objectif

- Comparer les « données » avec le « test »

Utilisation des spectres

Base de données

Complexité Temporelle

$O(nm^2)$

La complexité temporelle dépend également de la résolution de l'image acquise

Temps d'exécution

$0,1nm^2$ secondes

n : nombre d'individus dans la base de données

m : nombre de photos pour chaque individu

Méthodes de comparaison

par les moyennes : comparaison point à point

Calcul de la moyenne des valeurs des spectres « données »
Comparaison des moyennes obtenues avec le spectre « test »

Comparaison relative

- On détermine un écart de X % en plus ou en moins de la valeur en chaque point sur « données »
- On vérifie si la valeur de chaque point de « test » appartient à cet écart

Comparaison absolue

- On détermine un écart de plus ou moins x (où x est une constante) en chaque point sur « données »
- On vérifie si la valeur de chaque point de « test » appartient à cet écart

```
def Comparaison1(FFT1,FFT2):
    confiance1=0.3
    taux1=0
    for k in range(0,min(len(FFT1),len(FFT2))):
        for i in range(0,min(len(FFT1[k]),len(FFT2[k]))):
            if (FFT1[i][k]*(1-confiance1))<=(FFT2[i][k])<=(FFT1[i][k]*(1+confiance1)) or (FFT2[i][k]*(1-confiance1))<=(FFT1[i][k])<=(FFT2[i][k]*(1+confiance1)):
                taux1=taux1+1

    taux1=(taux1*100)/(min(len(FFT1),len(FFT2))*min(len(FFT1[0]),len(FFT2[0])))
    return(taux1)

```

```
def Comparaison2(FFT1,FFT2):
    confiance2=100
    taux2=0
    for k in range(0,min(len(FFT1),len(FFT2))):
        for i in range(0,min(len(FFT1[k]),len(FFT2[k]))):
            if FFT1[i][k]-confiance2<FFT2[i][k]<FFT1[i][k]+confiance2 or FFT2[i][k]-confiance2<FFT1[i][k]<FFT2[i][k]+confiance2:
                taux2=taux2+1

    taux2=(taux2*100)/(min(len(FFT1),len(FFT2))*min(len(FFT1[0]),len(FFT2[0])))
    return(taux2)

```

Méthodes de comparaison

par les intervalles : comparaison point à point

Pour « données » : détermination de l'intervalle
allant de la valeur minimale à la valeur maximale de chaque point
Pour « test » : on somme les distances respectives pour chaque intervalle

Extension naturelle

- Si la valeur de « test » appartient à l'intervalle : pas d'action
- Si la valeur de « test » n' appartient pas à l'intervalle : on la ramène à la borne la plus proche de l'intervalle

```
def ComparaisonDistance1(Liste,Intervalle):

    distance=0

    x=abs(len(Liste)-len(Intervalle[0]))//2

    for k in range(0,min(len(Liste),len(Intervalle[0]))):
        for i in range(0,min(len(Liste),len(Intervalle[0]))):
            if Intervalle[0][k][i]<=Liste[k][i]<=Intervalle[1][k][i]:
                distance=distance+0
            else:
                distance=distance+min(abs(Intervalle[0][k][i]-Liste[k]
[i]),abs(Intervalle[1][k][i]-Liste[k][i]))

    return(distance/
(min(len(Liste),len(Intervalle[0]))*min(len(Liste),len(Intervalle[0])))
```

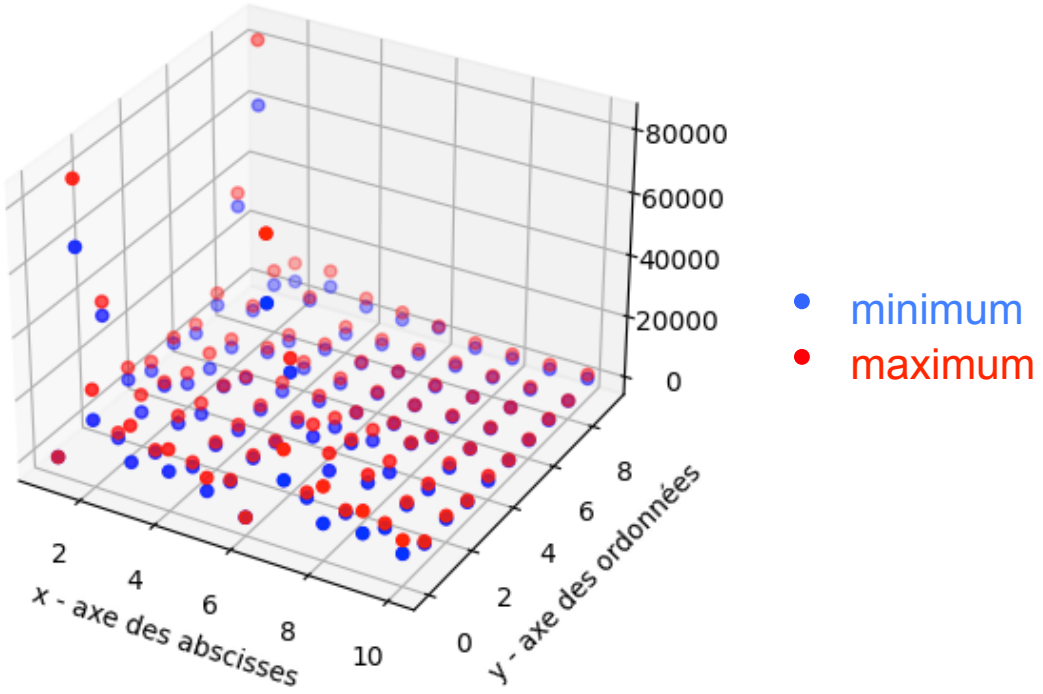
Méthode d'Hausdorff

- On ramène systématiquement la valeur « test » à la borne la plus éloignée de l'intervalle

```
def ComparaisonDistance2(Liste,Intervalle):

    distance=0

    for k in range(0,min(len(Liste),len(Intervalle[0]))):
        for i in range(0,min(len(Liste[0]),len(Intervalle[0][0]))):
            distance=distance+max(abs(Intervalle[0][k][i]-Liste[k]
[i]),abs(Intervalle[1][k][i]-Liste[k][i]))

    return(distance/
(min(len(Liste),len(Intervalle[0]))*min(len(Liste),len(Intervalle[0])))
```

Méthodes de comparaison

par les intervalles : comparaison point à point



Spectre des intervalles

Méthodes de comparaison

Résultats

Comparaison relative

Taux de réussite : 75 % avec X = 30

Comparaison absolue

Taux de réussite : 44 % avec x = 100
(pondération des valeurs)

Extension naturelle

Taux de réussite : 89 %

Méthode d'Hausdorff

Taux de réussite : 48 %

Méthodes de comparaison

Complexité

Complexité Temporelle

O(n)

Temps d'exécution

Traitement photo = 0,3 secondes + Temps de comparaison ≤ n (0,001 secondes)

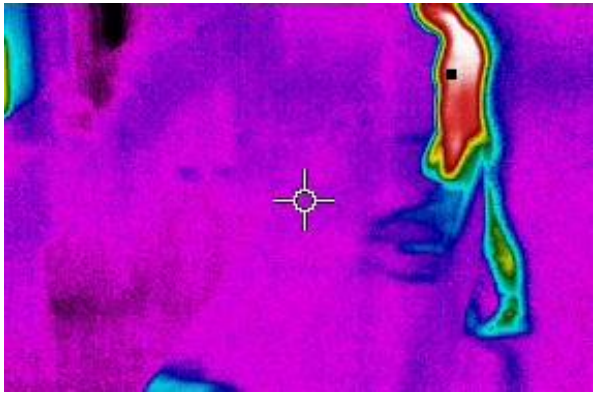
(n : nombre d'individus dans la base de données)

Application Pratique

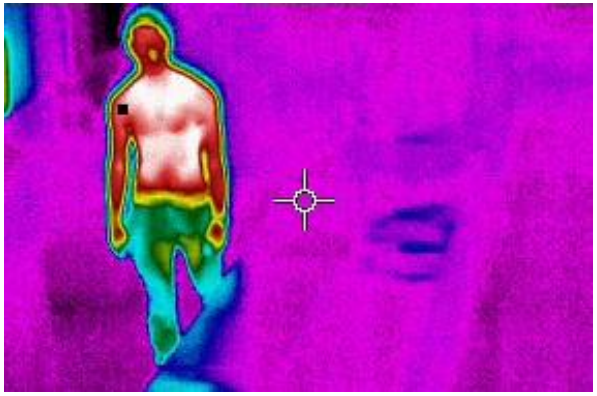
sécurisation piscine publique

Suivi en temps réel d'un individu par rapport au bassin

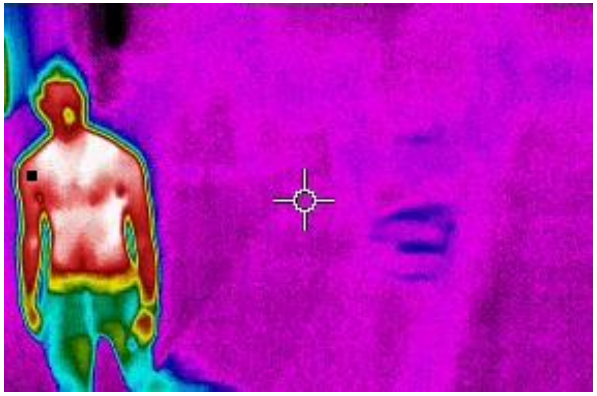
grâce au repère…



… carré noir



position connue…



… en moins de 0,5 s

Application Pratique

sécurisation piscine publique

Suivi en temps réel des usagers par rapport au bassin

- La signature thermique permet de suivre un ou plusieurs individus
- Si plusieurs individus rentrent dans le champ de la caméra, le suivi reste possible
- Si les trajectoires de ces individus se croisent, on risque de perdre le suivi d'un ou plusieurs individus
- Pour conserver un suivi permanent, il faut mettre en œuvre au moins 3 caméras
- Le suivi est possible jour et nuit

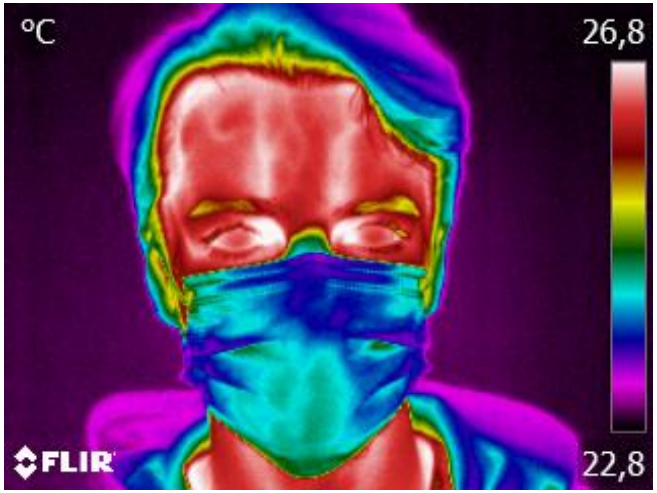
Limites et Optimisations

Limites

- Lunettes
- Masque

Optimisation

- Classe de réjection
- Autres méthode d’acquisition



Limites et Optimisations

Acquisition Video

Limites

- Echelle de température non linéaire
- Différence de 3° (moyenne) par pixel entre les valeurs du fichier CSV et celles de l'échelle de la photo

Annexes

- Programme Reconnaissance Faciale
- Programme de suivi vidéo

ANNEXE I

```python
import cv2
import os
import matplotlib.pyplot as plt
import numpy as np
import csv
from PIL import Image
import time

#-------------------------------------------------------------------------------
def Data():
    DATA=[]
    Name=[]

    for filename in os.listdir("C:/Users/Siena PAGE/Desktop/TIPE_BDD"):
        Name.append(filename)

    for k in range(0,len(Name)):
        DATA.append([])

    count1=0
    for i in Name:
        path="C:/Users/Siena PAGE/Desktop/TIPE_BDD"
        path=('/'.join(("C:/Users/Siena PAGE/Desktop/TIPE_BDD",'%s')) % i)

        count2=0
        for filename in os.listdir(path):
            type=filename.split('.')
            if type[1]=='jpg':
                count2=count2+1

        for k in range(1,count2):
            PhotoCsv=[]
            PhotoCsv.append(Image.open('/'.join((path, ''.join(('%s','%d.jpg')) % (i,k)))))
            PhotoCsv.append(open('/'.join((path, ''.join(('%s','%d.csv')) % (i,k)))))
            DATA[count1].append(PhotoCsv)

        DATA[count1].append(Name[count1])
        count1=count1+1
    return(DATA)

#-------------------------------------------------------------------------------
def Test():
    Test=[]
    Name=[]

    for filename in os.listdir("C:/Users/Siena PAGE/Desktop/TIPE_BDD"):
        Name.append(filename)

    for k in range(0,len(Name)):
        Test.append([])

```

ANNEXE I

```
    count1=0
    for i in Name:
        path="C:/Users/Siena PAGE/Desktop/TIPE_BDD"
        path=('/'.join(("C:/Users/Siena PAGE/Desktop/TIPE_BDD",'%s')) % i)

        count2=0
        for filename in os.listdir(path):
            type=filename.split('.')
            if type[1]=='jpg':
                count2=count2+1

        PhotoCsv=[]
        PhotoCsv.append(Image.open('/'.join((path, ''.join(('%s','%d.jpg')) % (i,count2)))))
        PhotoCsv.append(open('/'.join((path, ''.join(('%s','%d.csv')) % (i,count2)))))
        Test[count1].append(PhotoCsv)

        Test[count1].append(Name[count1])
        count1=count1+1
    return(Test)

#-------------------------------------------------------------------------------
def Taille(ImageColor):
    r=90
    g=140
    b=80

    ky=130
    YHaut=[]
    YBas=[]
    for m in range(0,13,5):
        yh=0
        yb=0
        for j in range(0,240):
            if ImageColor.getpixel((ky+m,j))[0]>r and ImageColor.getpixel((ky+m,j))[1]<g and ImageColor.getpixel((ky+m,j))[2]<b:
                if yh==0:
                    yh=j
        YHaut.append(yh)
        h=240
        for i in range(0,240):
            h=h-1
            if ImageColor.getpixel((ky+m,h))[0]>r and ImageColor.getpixel((ky+m,h))[1]<g and ImageColor.getpixel((ky+m,h))[2]<b:
                if yb==0:
                    yb=h
        YBas.append(yb)

    yh=YHaut[0]
    yb=YBas[0]
    for i in range(1,len(YHaut)):
        if yh>YHaut[i]:
            yh=YHaut[i]
        if yb<YBas[i]:
            yb=YBas[i]

    if (yb-yh)%2==1:
        yb=yb+1
```

ANNEXE I

```
    kx=90
    Xdroite=[]
    Xgauche=[]
    for m in range(0,13,5):
        xd=0
        xg=0
        for j in range(0,320):
            if ImageColor.getpixel((j,kx+m))[0]>r and ImageColor.getpixel((j,kx+m))[1]<g and ImageColor.getpixel((j,kx+m))[2]<b:
                if xg==0:
                    xg=j
        Xgauche.append(xg)
        h=290
        for i in range(0,290):
            h=h-1
            if ImageColor.getpixel((h,kx+m))[0]>r and ImageColor.getpixel((h,kx+m))[1]<g and ImageColor.getpixel((h,kx+m))[2]<b:
                if xd==0:
                    xd=h
        Xdroite.append(xd)

    xd=Xgauche[0]
    xg=Xdroite[0]
    for i in range(1,len(Xdroite)):
        if xd>Xgauche[i]:
            xd=Xgauche[i]
        if xg<Xdroite[i]:
            xg=Xdroite[i]

    if (xd-xg)%2==1:
        xd=xd+1

    centre=160-(xg+(xd-xg)//2)
    return(yh,yb,centre)

#-------------------------------------------------------------------------------
def crop(Image):
    yh,yb,decalage=Taille(Image)
    box=(160-decalage-(yb-yh)//2,yh,160-decalage+(yb-yh)//2,yb)
    ImageCropped=Image.crop(box)
    return(ImageCropped)

#-------------------------------------------------------------------------------
def Fourier(Tableau):
    T=[]
    a=0
    for row in Tableau:
        a=a+1
        F=[]
        if len(row[0].split(';'))==1:
            PartieEntiereRelou=row[0]
        else:
            for k in range(0,len(row[0].split(';'))-1):
                F.append(float(row[0].split(';')[k]))
        PartieFractionnaireRelou=(row[1].split(';'))[0]
        F.append(float('.'.join((PartieEntiereRelou,PartieFractionnaireRelou))))

```

ANNEXE I

```
        k=0
        while len(F)<320-len(row[-1].split(';')):
            k=k+1
            PartieEntiereEntier=[]
            if len(row[k].split(';'))==2:
                PartieEntiere=(row[k].split(';'))[1]
            else:
                PartieEntiere=(row[k].split(';'))[-1]
                entier=0
                while len(row[k].split(';'))-entier>2:
                    entier=entier+1
                    PartieEntiereEntier.append((row[k].split(';'))[entier])

            PartieFractionnaire=(row[k+1].split(';'))[0]

            if len(PartieEntiereEntier)!=0:
                for i in range(0,len(PartieEntiereEntier)):
                    F.append(float(PartieEntiereEntier[i]))

            if float('.'.join((PartieEntiere,PartieFractionnaire)))<10 or float('.'.join((PartieEntiere,PartieFractionnaire)))>40:
                print(float('.'.join((PartieEntiere,PartieFractionnaire))))


            F.append(float('.'.join((PartieEntiere,PartieFractionnaire))))

        PartieEntiereRelou2=(row[-2].split(';'))[-1]
        if len(row[-1].split(';'))!=1:
            F.append(float('.'.join((PartieEntiereRelou2,row[-1].split(';')[0]))))
            for i in range(1,len(row[-1].split(';'))):
                F.append(float(row[-1].split(';')[i]))
        else:
            PartieFractionnaireRelou2=row[-1]
        F.append(float('.'.join((PartieEntiereRelou2,PartieFractionnaireRelou2))))
        T.append(F)

    return(T)
#-------------------------------------------------------------------------------
def CSVcrop(T,Image):
    Tcrop=[]

    yh,yb,decalage=Taille(Image)
    box=(160-decalage-(yb-yh)//2,yh,160-decalage+(yb-yh)//2,yb)

    for i in range(box[1],box[3]):
        TcropTempo=[]
        for k in range(box[0],box[2]):
            TcropTempo.append(T[i][k])
        Tcrop.append(TcropTempo)
    return(Tcrop)

```

ANNEXE I

```
#-------------------------------------------------------------------------------
def Moyenne(Matrice):
    somme=0
    for i in range(0,len(Matrice)):
        for k in range(0,len(Matrice[i])):
            somme=somme+Matrice[i][k]

    moy=somme/(len(Matrice)*len(Matrice[0]))

    for i in range(0,len(Matrice)):
        for k in range(0,len(Matrice[i])):
            Matrice[i][k]=Matrice[i][k]-moy

#-------------------------------------------------------------------------------
def MoyenneFFT(Liste):
    SommeFFT=[]
    for k in range(0,len(Liste[0])):
        SommeFFT.append([0])
        for i in range(0,len(Liste[0][0])-1):
            SommeFFT[k].append(0)

    for k in range(0,len(Liste[0])):
        for i in range(0,len(Liste[0][0])):
            for j in range(0,len(Liste)):
                SommeFFT[k][i]=SommeFFT[k][i]+Liste[j][k][i]

    for k in range(0,len(SommeFFT)):
        for i in range(0,len(SommeFFT[0])):
            SommeFFT[k][i]=SommeFFT[k][i]/len(Liste)
    return(SommeFFT)

#-------------------------------------------------------------------------------
def Distance(Liste):
    ListeMin=[]

    for k in range(0,len(Liste[0])):
        ListeMin.append([0])
        for i in range(0,len(Liste[0][0])-1):
            ListeMin[k].append(0)

    for k in range(0,len(Liste[0])):
        for i in range(0,len(Liste[0][0])):
            min=Liste[0][k][i]
            for j in range(0,len(Liste)):
                if Liste[j][k][i]<min:
                    min=Liste[j][k][i]
            ListeMin[k][i]=min

    ListeMax=[]

```

ANNEXE I

```
    for k in range(0,len(Liste[0])):
        ListeMax.append([0])
        for i in range(0,len(Liste[0][0])-1):
            ListeMax[k].append(0)

    for k in range(0,len(Liste[0])):
        for i in range(0,len(Liste[0][0])):
            max=Liste[0][0][0]
            for j in range(0,len(Liste)):
                if Liste[j][k][i]>max:
                    max=Liste[j][k][i]
            ListeMax[k][i]=max

    return(ListeMin,ListeMax)

#-------------------------------------------------------------------------------
def ReduceFFT1(FFT):
    FFT=np.fft.fft2(FFT)

    FFTreduce=[]

    FFT1=FFT[0:5]
    for k in range(0,len(FFT1)):
        FFTreduce.append((FFT[k])[:5])

    FFT2=FFT[0:5]
    for k in range(0,len(FFT2)):
        FFTreduce[k]=np.concatenate((FFTreduce[k],FFT[k][len(FFT)-5:len(FFT)]))

    FFT3=FFT[len(FFT)-5:]
    for k in range(0,len(FFT3)):
        FFTreduce.append((FFT[k])[:5])

    FFT4=FFT[len(FFT)-5:]
    for k in range(5,5+len(FFT4)):
        FFTreduce[k]=np.concatenate((FFTreduce[k],FFT[k][len(FFT)-5:len(FFT)]))

    FFT=np.array(FFTreduce)
    FFT=abs(FFT)
    return(FFT)

#-------------------------------------------------------------------------------
def ReduceFFT2(FFT):
    FFT=np.fft.fft2(FFT)
    FFT=np.array(FFT)
    FFT=abs(FFT)
    return(FFT)

```

ANNEXE I

```
#-------------------------------------------------------------------------------
def homogenisation(Liste):
    taille=len(Liste[0])

    ListeHomo=[]

    for k in range(1,len(Liste)):
        print('oui',len(Liste[k]))
        if taille>len(Liste[k]):
            taille=len(Liste[k])

    for k in range(0,len(Liste)-1):
        ListeHomo.append(np.zeros([taille,taille]))

    for k in range(0,len(Liste)-1):
        print('1',len(Liste[k]),len(Liste[k][0]))

        if (len(Liste[k])-taille)%2==0:
            x=(len(Liste[k])-taille)//2
        else:
            x=(len(Liste[k][i])-taille+1)//2

        for i in range(0,taille):
            if (len(Liste[k])-taille)%2==0:
                ListeHomo[k][i]=(Liste[k][i+x])[x:len(Liste[k][i])-x]
            else:
                ListeHomo[k][i]=(Liste[k][i+x])[x:len(Liste[k][i])-x+1]

        print('2',len(ListeHomo[k]),len(ListeHomo[k][0]))

    return(ListeHomo)

#-------------------------------------------------------------------------------
def Comparaison1(FFT1,FFT2):
    confiance1=0.3
    taux1=0
    for k in range(0,min(len(FFT1),len(FFT2))):
        for i in range(0,min(len(FFT1[k]),len(FFT2[k]))):
            if (FFT1[i][k]*(1-confiance1))<(FFT2[i][k])<=(FFT1[i][k]*(1+confiance1)) or (FFT2[i][k]*(1-confiance1))<=(FFT1[i][k])<=(FFT2[i][k]*(1+conf
                taux1=taux1+1

    taux1=(taux1*100)/(min(len(FFT1),len(FFT2))*min(len(FFT1[0]),len(FFT2[0])))
    return(taux1)

#-------------------------------------------------------------------------------
def Comparaison2(FFT1,FFT2):
    confiance2=100
    taux2=0
    for k in range(0,min(len(FFT1),len(FFT2))):
        for i in range(0,min(len(FFT1[k]),len(FFT2[k]))):
            if FFT1[i][k]-confiance2<FFT2[i][k]<FFT1[i][k]+confiance2 or FFT2[i][k]-confiance2<FFT1[i][k]<FFT2[i][k]+confiance2:
                taux2=taux2+1

    taux2=(taux2*100)/(min(len(FFT1),len(FFT2))*min(len(FFT1[0]),len(FFT2[0])))
    return(taux2)

```

ANNEXE I

```
#-------------------------------------------------------------------------------
def ComparaisonDistance1(Liste,Intervalle):

    distance=0

    x=abs(len(Liste)-len(Intervalle[0]))//2

    for k in range(0,min(len(Liste),len(Intervalle[0]))):
        for i in range(0,min(len(Liste),len(Intervalle[0]))):
            if Intervalle[0][k][i]<=Liste[k][i]<=Intervalle[1][k][i]:
                distance=distance+0
            else:
                distance=distance+min(abs(Intervalle[0][k][i]-Liste[k][i]),abs(Intervalle[1][k][i]-Liste[k][i]))

    return(distance/(min(len(Liste),len(Intervalle[0]))*min(len(Liste),len(Intervalle[0]))))

#-------------------------------------------------------------------------------
def ComparaisonDistance2(Liste,Intervalle):

    distance=0

    for k in range(0,min(len(Liste),len(Intervalle[0]))):
        for i in range(0,min(len(Liste[0]),len(Intervalle[0][0]))):
            distance=distance+max(abs(Intervalle[0][k][i]-Liste[k][i]),abs(Intervalle[1][k][i]-Liste[k][i]))

    return(distance/(min(len(Liste),len(Intervalle[0]))*min(len(Liste),len(Intervalle[0]))))

#-------------------------------------------------------------------------------
def Axes(FFT):
    X=[]
    Y=[]
    XX=[]
    YY=[]

    count=0
    for row in FFT:
        count=count+1
        for k in range(0,len(row)):
            X.append(count)
            Y.append(k)
        XX=np.concatenate((XX,X))
        YY=np.concatenate((YY,Y))
        X=[]
        Y=[]
    return(XX,YY)

#-------------------------------------------------------------------------------
def graph(courbe):
    ax = plt.axes(projection='3d')

    x,y=Axes(courbe)
    ax.scatter(x, y, courbe, c='blue')

    plt.xlabel('x - axe des abscisses ')
    plt.ylabel('y - axe des ordonnées')
    plt.grid()
    ax.set_title('FFT')
    plt.show()

```

ANNEXE I

```
#-------------------------------------------------------------------------------
def CSVManuel(Image):

    Tmax=33.3
    Tmin=13.0

    Image=Image.crop((15,25,Image.size[0],Image.size[1]-27))

    GradT=(Tmax-Tmin)/180

    T=[]

    startx=0
    endx=Image.size[0]
    starty=0
    endy=Image.size[1]

    echelle=[]
    for y in range(5,183):
        echelleK=[0,0,0]
        for x in range(292,298):
            for k in range(0,3):
                echelleK[k]=echelleK[k]+Image.getpixel((x,y))[k]
        for k in range(0,3):
            echelleK[k]=int(echelleK[k]//5)
        echelle.append(tuple(echelleK))

    Croix=[]
    for k in range(130,140):
        for i in range(94,97):
            Croix.append((k,i))

    for k in range(151,160):
        for i in range(94,97):
            Croix.append((k,i))

    for k in range(144,147):
        for i in range(101,110):
            Croix.append((k,i))

    for k in range(144,147):
        for i in range(80,90):
            Croix.append((k,i))

    for k in range(142,149):
        for i in range(90,92):
            Croix.append((k,i))

    for k in range(142,149):
        for i in range(99,101):
            Croix.append((k,i))

    for k in range(149,151):
        for i in range(92,99):
            Croix.append((k,i))

    for k in range(140,142):
        for i in range(92,99):
            Croix.append((k,i))

```

ANNEXE I

```
    Croix.append((141,91))
    Croix.append((149,91))
    Croix.append((142,92))
    Croix.append((148,92))
    Croix.append((142,98))
    Croix.append((148,98))
    Croix.append((141,99))
    Croix.append((149,99))
    for y in range(starty,endy):
        Tx=[]
        for x in range(startx,endx):
            if (x,y) not in Croix:
                value=True
                value2=True
                f=0
                while value2:
                    f=f+10
                    for i in range(0,178):
                        if (echelle[i][0]-f)<Image.getpixel((x,y))[0]<(echelle[i][0]+f) and (echelle[i][1]-f)<Image.getpixel((x,y))
[1]<(echelle[i][1]+f) and (echelle[i][2]-f)<Image.getpixel((x,y))[2]<(echelle[i][2]+f) and value:
                            Tx.append(Tmin+(180-i)*GradT)
                            Image.putpixel((x,y),echelle[i])
                            value=False
                            value2=False
            else:
                Tx.append(Tx[-1])
                Image.putpixel((x,y),Image.getpixel((x-1,y)))

        T.append(Tx)
    return(T)

#-------------------------------------------------------------------------------
def CréationListe(Image,Fichier):
    Tableau=csv.reader(Fichier)
    ImageCropped=crop(Image)
    CSV=Fourier(Tableau)
    T=CSVcrop(CSV,Image)
    Moyenne(T)
    return(T)

#-------------------------------------------------------------------------------
def BaseDeDonnée1(Photo):
    BDD=[]
    for k in range(0,len(Photo)):
        MoyenneK=[]
        for i in range(0,len(Photo[k])-1):
            MoyenneK.append(ReduceFFT1(CréationListe(Photo[k][i][0],Photo[k][i][1])))
        MoyenneK=MoyenneFFT(MoyenneK)
        BDD.append([Photo[k][-1],MoyenneK])

    return(BDD)
```

ANNEXE I

```
#-------------------------------------------------------------------------------
def BaseDeDonnée2(Photo,PhotoTest):
    BDD=[]
    for k in range(0,len(Photo)):
        ListeDistance=[]
        for i in range(0,len(Photo[k])-1):
            ListeDistance.append(CréationListe(Photo[k][i][0],Photo[k][i][1]))
        for i in range(0,len(ListeDistance)):
            ListeDistance[i]=ReduceFFT1(ListeDistance[i])
        ListeDistanceBDD=Distance(ListeDistance)
        BDD.append([Photo[k][-1],ListeDistanceBDD])

    return(BDD)

##
PhotoTest=Test()
Photo=Data()


BDD=BaseDeDonnée1(Photo)

count=0
for i in range(0,len(PhotoTest)):
    TAUX=[]
    FFTtest=ReduceFFT1(CréationListe(PhotoTest[i][0][0],PhotoTest[i][0][1]))
    for k in range(0,len(BDD)):
        TAUX.append([Comparaison1(BDD[k][1],FFTtest),BDD[k][0]])
    ressemblance=0
    gagnant=0

    for k in range(0,len(TAUX)):
        if TAUX[k][0]>=ressemblance:
            ressemblance=TAUX[k][0]
            gagnant=k

    difference=ressemblance
    for k in range(0,len(TAUX)):
        if k!=gagnant:
            if ressemblance-TAUX[k][0]<difference:
                difference=ressemblance-TAUX[k][0]
                posdiff=k

    if TAUX[gagnant][1]==PhotoTest[i][1]:

        count=count+1

```

ANNEXE I

```
583
584 ##
585 TOTAL=0
586
587 for lol in range(0,5):
588     DATA=[]
589     Name=[]
590
591     for filename in os.listdir("C:/Users/Siena PAGE/Desktop/TIPE_BDD"):
592         Name.append(filename)
593
594     for k in range(0,len(Name)):
595         DATA.append([])
596
597     count1=0
598     for i in Name:
599         path="C:/Users/Siena PAGE/Desktop/TIPE_BDD"
600         path=('/'.join(("C:/Users/Siena PAGE/Desktop/TIPE_BDD",'%s')) % i)
601
602         count2=0
603         for filename in os.listdir(path):
604             type=filename.split('.')
605             if type[1]=='jpg':
606                 count2=count2+1
607
608         for k in range(1,count2+1):
609             if k!=lol+1:
610                 PhotoCsv=[]
611                 PhotoCsv.append(Image.open('/'.join((path, ''.join(('%s','%d.jpg')) % (i,k)))))
612                 PhotoCsv.append(open('/'.join((path, ''.join(('%s','%d.csv')) % (i,k)))))
613                 DATA[count1].append(PhotoCsv)
614
615         DATA[count1].append(Name[count1])
616         count1=count1+1
617
618     Test=[]
619
620     for k in range(0,len(Name)):
621         Test.append([])
622
623     count1=0
624     for i in Name:
625         path="C:/Users/Siena PAGE/Desktop/TIPE_BDD"
626         path=('/'.join(("C:/Users/Siena PAGE/Desktop/TIPE_BDD",'%s')) % i)
627
628         count2=lol+1
629
630         PhotoCsv=[]
631         PhotoCsv.append(Image.open('/'.join((path, ''.join(('%s','%d.jpg')) % (i,count2)))))
632         PhotoCsv.append(open('/'.join((path, ''.join(('%s','%d.csv')) % (i,count2)))))
633         Test[count1].append(PhotoCsv)
634
635         Test[count1].append(Name[count1])
636         count1=count1+1
637
638     Photo=DATA
639     PhotoTest=Test
640
```

ANNEXE I

```
    BDD=BaseDeDonnée2(Photo,PhotoTest)

    count=0
    for i in range(0,len(PhotoTest)):
        TAUX=[]
        FFTtest=ReduceFFT1(CréationListe(PhotoTest[i][0][0],PhotoTest[i][0][1]))
        for k in range(0,len(BDD)):
            TAUX.append([ComparaisonDistance2(FFTtest,BDD[k][1]),BDD[k][0]])
        distance=TAUX[i][0]
        gagnant=0

        for k in range(0,len(TAUX)):
            if TAUX[k][0]<=distance:
                distance=TAUX[k][0]
                gagnant=k

        posdiff=0

        difference=TAUX[k][0]
        for k in range(0,len(TAUX)):
            if k!=gagnant:
                if TAUX[k][0]-distance<difference:
                    difference=TAUX[k][0]-distance
                    posdiff=k

        if TAUX[gagnant][1]==PhotoTest[i][1]:
            count=count+1

    TOTAL=TOTAL+count

print('Le nombre de bonne reconnaissance est',TOTAL,"pour 75 tests")
print("Soit un taux de",TOTAL*100/75,"%")


##
TOTAL=0
for lol in range(0,5):
    DATA=[]
    Name=[]

    for filename in os.listdir("C:/Users/Siena PAGE/Desktop/TIPE_BDD"):
        Name.append(filename)

    for k in range(0,len(Name)):
        DATA.append([])

    count1=0
    for i in Name:
        path="C:/Users/Siena PAGE/Desktop/TIPE_BDD"
        path=('/'.join(("C:/Users/Siena PAGE/Desktop/TIPE_BDD",'%s')) % i)

        count2=0
        for filename in os.listdir(path):
            type=filename.split('.')
            if type[1]=='jpg':
                count2=count2+1
```

ANNEXE I

```
        for k in range(1,count2+1):
            if k!=lol+1:
                PhotoCsv=[]
                PhotoCsv.append(Image.open('/'.join((path, ''.join(('%s','%d.jpg')) % (i,k)))))
                PhotoCsv.append(open('/'.join((path, ''.join(('%s','%d.csv')) % (i,k)))))
                DATA[count1].append(PhotoCsv)

        DATA[count1].append(Name[count1])
        count1=count1+1

    Test=[]

    for k in range(0,len(Name)):
        Test.append([])

    count1=0
    for i in Name:
        path="C:/Users/Siena PAGE/Desktop/TIPE_BDD"
        path=('/'.join(("C:/Users/Siena PAGE/Desktop/TIPE_BDD",'%s')) % i)
        count2=lol+1
        PhotoCsv=[]
        PhotoCsv.append(Image.open('/'.join((path, ''.join(('%s','%d.jpg')) % (i,count2)))))
        PhotoCsv.append(open('/'.join((path, ''.join(('%s','%d.csv')) % (i,count2)))))
        Test[count1].append(PhotoCsv)

        Test[count1].append(Name[count1])
        count1=count1+1

    Photo=DATA
    PhotoTest=Test
    BDD=BaseDeDonnée1(Photo)

    count=0
    for i in range(0,len(PhotoTest)):
        TAUX=[]
        FFTtest=ReduceFFT1(CréationListe(PhotoTest[i][0][0],PhotoTest[i][0][1]))
        for k in range(0,len(BDD)):
            TAUX.append([Comparaison1(BDD[k][1],FFTtest),BDD[k][0]])
        ressemblance=0
        gagnant=0

        for k in range(0,len(TAUX)):
            if TAUX[k][0]>=ressemblance:
                ressemblance=TAUX[k][0]
                gagnant=k

        difference=ressemblance
        for k in range(0,len(TAUX)):
            if k!=gagnant:
                if ressemblance-TAUX[k][0]<difference:
                    difference=ressemblance-TAUX[k][0]
                    posdiff=k

        if TAUX[gagnant][1]==PhotoTest[i][1]:
            count=count+1
    TOTAL=TOTAL+count

print('Le nombre de bonne reconnaissance est',TOTAL,"pour 75 tests")
print("Soit un taux de",TOTAL*100/75,"%")
```

ANNEXE II

```python
import cv2
import os
from PIL import Image
import numpy as np
import PIL
import time

def croix():
    Croix=[]
    for k in range(130,140):
        for i in range(94,97):
            Croix.append((k,i))

    for k in range(151,160):
        for i in range(94,97):
            Croix.append((k,i))

    for k in range(144,147):
        for i in range(101,110):
            Croix.append((k,i))

    for k in range(144,147):
        for i in range(80,90):
            Croix.append((k,i))

    for k in range(142,149):
        for i in range(90,92):
            Croix.append((k,i))

    for k in range(142,149):
        for i in range(99,101):
            Croix.append((k,i))

    for k in range(149,151):
        for i in range(92,99):
            Croix.append((k,i))

    for k in range(140,142):
        for i in range(92,99):
            Croix.append((k,i))

    Croix.append((141,91))
    Croix.append((149,91))
    Croix.append((142,92))
    Croix.append((148,92))
    Croix.append((142,98))
    Croix.append((148,98))
    Croix.append((141,99))
    Croix.append((149,99))

    return(Croix)

```

ANNEXE II

```
video='Desktop/TipeNelloVideo.mp4'
path_output_dir='Desktop/VideoTestTipe'

vidcap = cv2.VideoCapture(video)

fps=vidcap.get(cv2.CAP_PROP_FPS)

Croix=croix()

start_time = time.time()

nbImageSeconde=2

ListeImage=[]


count = 0
while vidcap.isOpened():
    success, image = vidcap.read()
    if success:
        cv2.imwrite(os.path.join(path_output_dir, '%d.png') % count, image)
        count=count+1
        if count%(30//nbImageSeconde)==0:
            ImageTest=Image.open("Desktop/videoTestTipe/%d.png" % count)
            ListeImage.append(ImageTest.crop((15,25,ImageTest.size[0]-20,ImageTest.size[1]-27)))

    else:
        break
cv2.destroyAllWindows()
vidcap.release()



for k in range(0,4):
    ListeImage.pop()

rayon=100

lastpos=(0,0)
ListePos=[]

```

ANNEXE II

```
for k in range(0,len(ListeImage)):
    pos=True
    image=ListeImage[k]

    if lastpos[0]-rayon//2>=0:
        xmin=lastpos[0]-rayon//2
    else:
        xmin=0

    if lastpos[0]+rayon//2<=image.size[0]:
        xmax=lastpos[0]+rayon//2
    else:
        xmax=image.size[0]


    if lastpos[1]-rayon//2>=0:
        ymin=lastpos[1]-rayon//2
    else:
        ymin=0

    if lastpos[1]+rayon//2<=image.size[1]:
        ymax=lastpos[1]+rayon//2
    else:
        ymax=image.size[1]


    if k==0:
        xmin=0
        ymin=0
        xmax=image.size[0]
        ymax=image.size[1]


    for x in range(xmin,xmax):
        for y in range(ymin,ymax):
            if (x,y) not in Croix:
                if 200<image.getpixel((x,y))[0]<260 and 40<image.getpixel((x,y))[1]<260 and 100<image.getpixel((x,y))[2]<260:
                    if pos==True:
                        pos=(x,y)
                        lastpos=pos
                        print((time.time() - start_time))
    ListePos.append(pos)


for k in range(0,len(ListePos)):
    for j in range(0,5):
        for i in range(0,5):
            if ListePos[k][0]+j>=ListeImage[k].size[0] or ListePos[k][1]+i>=ListeImage[k].size[1]:
                test=0
            else:
                ListeImage[k].putpixel((ListePos[k][0]+j,ListePos[k][1]+i),(0,0,0))

for i in range(0,len(ListeImage)):
    ListeImage[i].save("Desktop/videoTipeSave/%d.jpg" % i)

```