

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

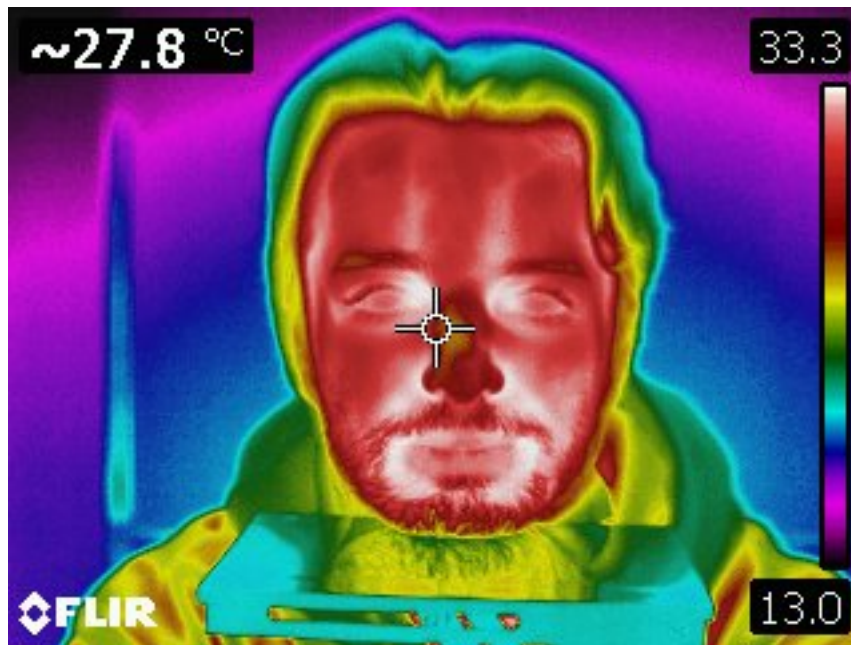
- Résolution IR : 320 x 240 pixels
- Plage de mesure des températures : -20 à 650°C
- Taux de rafraichissement : 60Hz
- Sensibilité thermique : < 0,05°C

Fonctionnement

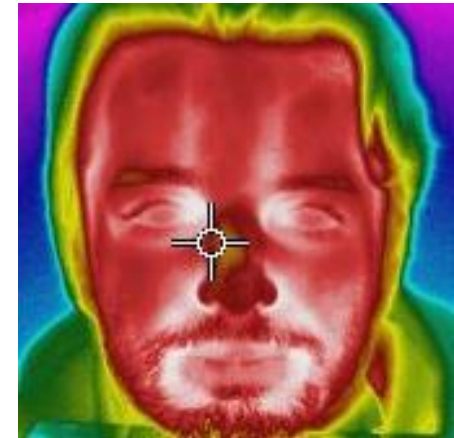
- Capteur de flux thermique
- Loi de Stefan
pour un corps gris d'émissivité ε :
 $\Phi = \varepsilon \sigma T^4$

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

```
1 def CSVcrop(T, Image):
2     Tcrop=[]
3
4     yh,yb,decalage=Taille(Image)
5     box=(160-decalage-(yb-yh)//2,yh,160-decalage+(yb-yh)//2,yb)
6
7     for i in range(box[1],box[3]):
8         TcropTempo=[]
9         for k in range(box[0],box[2]):
10             TcropTempo.append(T[i][k])
11         Tcrop.append(TcropTempo)
12     return(Tcrop)
```

Mise en correspondance
du fichier de données

```
1 def Moyenne(Matrice):
2     somme=0
3     for i in range(0,len(Matrice)):
4         for k in range(0,len(Matrice[i])):
5             somme=somme+Matrice[i][k]
6
7     moy=somme/(len(Matrice)*len(Matrice[0]))
8
9     for i in range(0,len(Matrice)):
10        for k in range(0,len(Matrice[i])):
11            Matrice[i][k]=Matrice[i][k]-moy
12
```

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

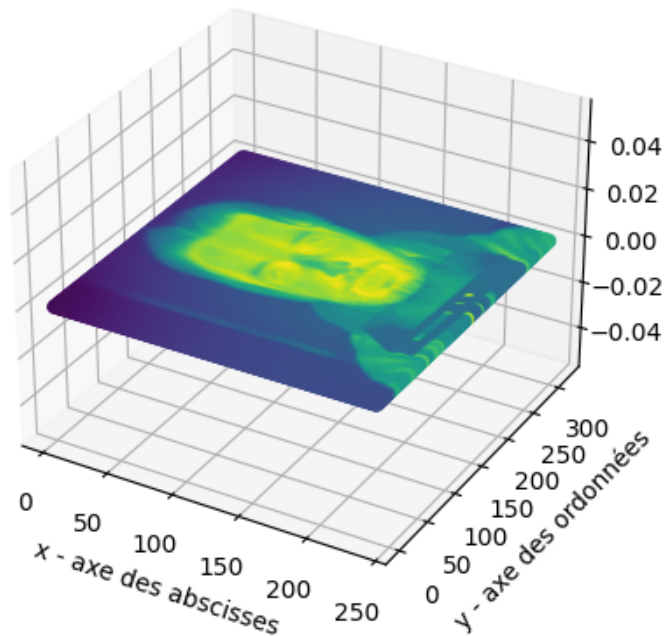
$$S(\sigma_x, \sigma_y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n) e^{-i2\pi(\frac{\sigma_x m}{M} + \frac{\sigma_y n}{N})}$$

Transformée de Fourier discrétisée (FTD)

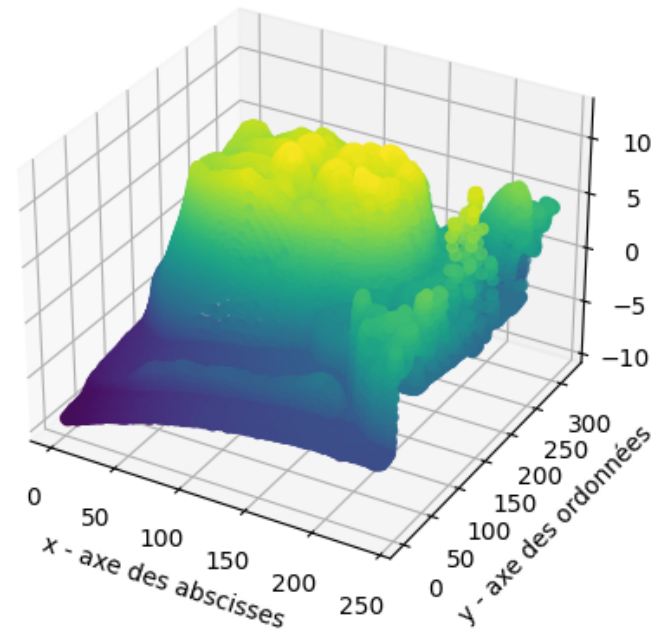
FFT : Algorithme de résolution de la Transformée de Fourier

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

Représentations Graphiques



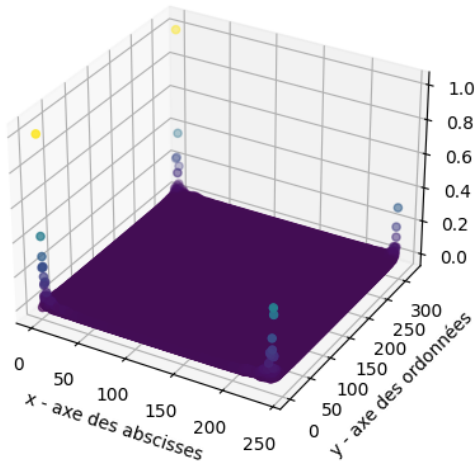
Représentation 2D
des températures



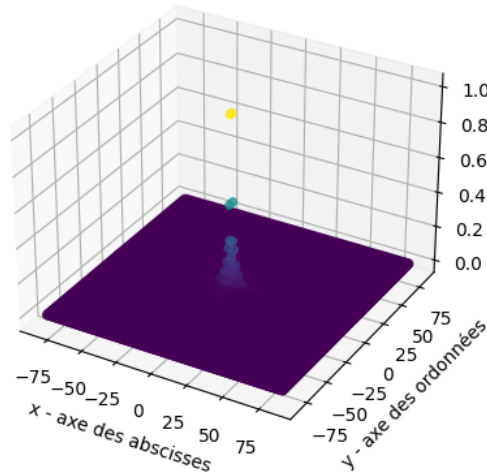
Représentation 3D
des températures

Traitements des spectres

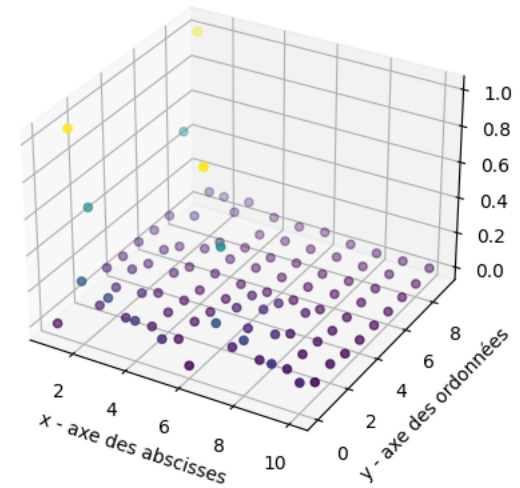
réduction des spectres :
conservation des valeurs critiques



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

0,1nm² 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

```
1 def Comparaison1(FFT1,FFT2):
2     confiance1=0.3
3     taux1=0
4     for k in range(0,min(len(FFT1),len(FFT2))):
5         for i in range(0,min(len(FFT1[k]),len(FFT2[k]))):
6             if (FFT1[i][k]*(1-confiance1))<=(FFT2[i][k])<=(FFT1[i]
7             [k]*(1+confiance1)) or (FFT2[i][k]*(1-confiance1))<=(FFT1[i][k])<=(FFT2[i]
8             [k]*(1+confiance1)):
9                 taux1=taux1+1
10
11     taux1=(taux1*100)/(min(len(FFT1), len(FFT2))*min(len(FFT1[0]), len(FFT2[0])))
12     return(taux1)
```

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

```
1 def Comparaison2(FFT1,FFT2):
2     confiance2=100
3     taux2=0
4     for k in range(0,min(len(FFT1),len(FFT2))):
5         for i in range(0,min(len(FFT1[k]),len(FFT2[k]))):
6             if FFT1[i][k]-confiance2<FFT2[i][k]<FFT1[i][k]+confiance2 or FFT2[i]
7             [k]-confiance2<FFT1[i][k]<FFT2[i][k]+confiance2:
8                 taux2=taux2+1
9
10     taux2=(taux2*100)/(min(len(FFT1), len(FFT2))*min(len(FFT1[0]), len(FFT2[0])))
11     return(taux2)
12
```

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

```
1 def ComparaisonDistance1(Liste, Intervalle):
2     distance=0
3
4     x=abs(len(Liste)-len(Intervalle[0]))//2
5
6     for k in range(0,min(len(Liste),len(Intervalle[0]))):
7         for i in range(0,min(len(Liste),len(Intervalle[0]))):
8             if Intervalle[0][k][i]<=Liste[k][i]<=Intervalle[1][k][i]:
9                 distance=distance+0
10            else:
11                distance=distance+min(abs(Intervalle[0][k][i]-Liste[k][i]),abs(Intervalle[1][k][i]-Liste[k][i]))
12
13     return(distance/
14           (min(len(Liste),len(Intervalle[0]))*min(len(Liste),len(Intervalle[0]))))
15
```

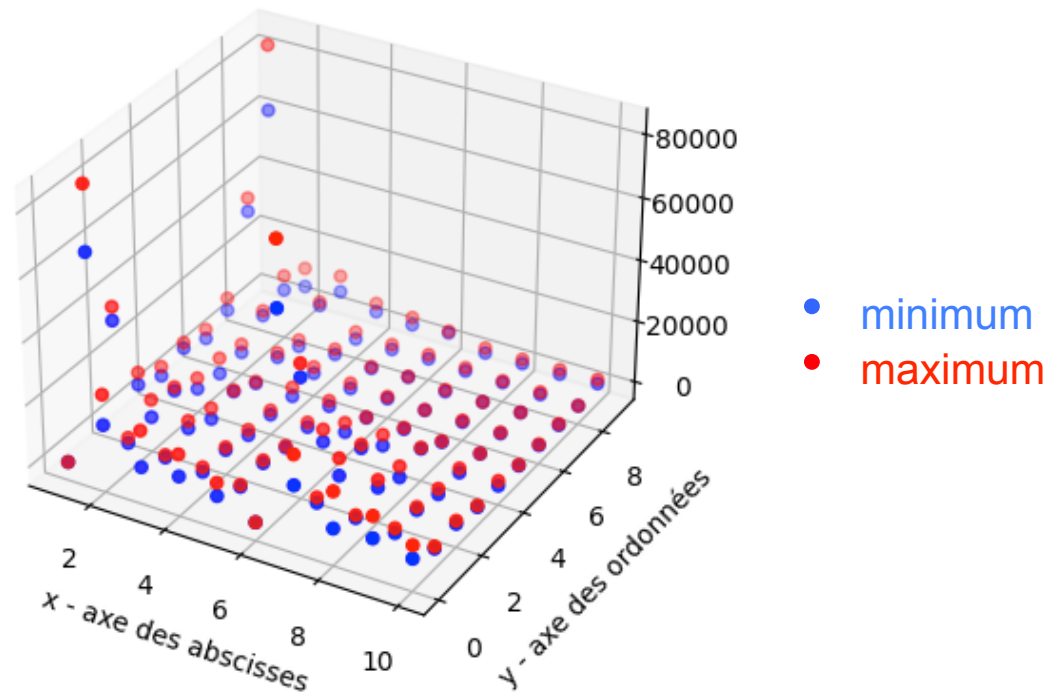
Méthode d'Hausdorff

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

```
1 def ComparaisonDistance2(Liste, Intervalle):
2     distance=0
3
4     for k in range(0,min(len(Liste),len(Intervalle[0]))):
5         for i in range(0,min(len(Liste[0]),len(Intervalle[0][0]))):
6             distance=distance+max(abs(Intervalle[0][k][i]-Liste[k][i]),abs(Intervalle[1][k][i]-Liste[k][i]))
7
8     return(distance/
9           (min(len(Liste),len(Intervalle[0]))*min(len(Liste),len(Intervalle[0]))))
10
11
12
```

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

$$\begin{array}{lcl} \text{Traitement photo} & + & \text{Temps de comparaison} \\ = 0,3 \text{ secondes} & & \leq n (0,001 \text{ secondes}) \end{array}$$

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

Application Pratique

sécurisation piscine publique

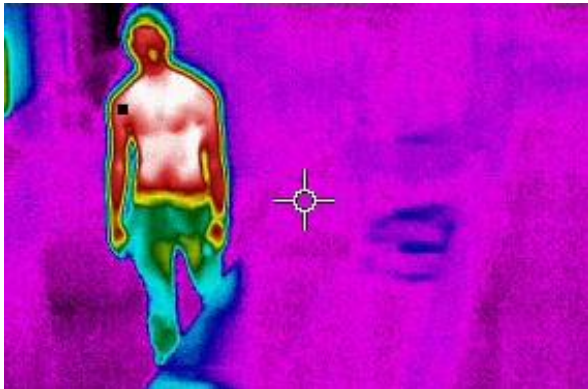
Suivi en temps réel d'un individu par rapport au bassin



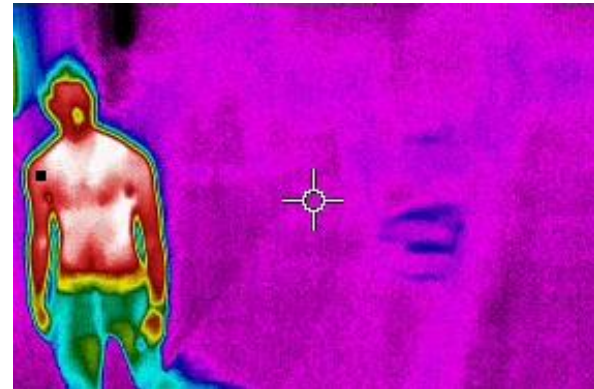
grâce au repère...



... 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

```
1 import cv2
2 import os
3 import matplotlib.pyplot as plt
4 import numpy as np
5 import csv
6 from PIL import Image
7 import time
8
9 #-----
10 def Data():
11     DATA=[]
12     Name=[]
13
14     for filename in os.listdir("C:/Users/Siena PAGE/Desktop/TIPE_BDD"):
15         Name.append(filename)
16
17     for k in range(0,len(Name)):
18         DATA.append([])
19
20     count1=0
21     for i in Name:
22         path="C:/Users/Siena PAGE/Desktop/TIPE_BDD"
23         path=os.path.join(path,i)
24
25         count2=0
26         for filename in os.listdir(path):
27             type=filename.split('.')
28             if type[1]=='jpg':
29                 count2=count2+1
30
31         for k in range(1,count2):
32             PhotoCsv=[]
33             PhotoCsv.append(Image.open(os.path.join(path, os.path.join(i, '%d.jpg' % (i,k)))))
34             PhotoCsv.append(open(os.path.join(path, os.path.join(i, '%d.csv' % (i,k)))))
35             DATA[count1].append(PhotoCsv)
36
37         DATA[count1].append(Name[count1])
38         count1=count1+1
39     return(DATA)
40
41 #-----
42 def Test():
43     Test=[]
44     Name=[]
45
46     for filename in os.listdir("C:/Users/Siena PAGE/Desktop/TIPE_BDD"):
47         Name.append(filename)
48
49     for k in range(0,len(Name)):
50         Test.append([])
51
```


ANNEXE I

```
52 count1=0
53 for i in Name:
54     path="C:/Users/Siena PAGE/Desktop/TIPE_BDD"
55     path=('/' .join(("C:/Users/Siena PAGE/Desktop/TIPE_BDD", '%s')) % i)
56
57     count2=0
58     for filename in os.listdir(path):
59         type=filename.split('.')
60         if type[1]=='jpg':
61             count2=count2+1
62
63     PhotoCsv=[]
64     PhotoCsv.append(Image.open('/' .join((path, '/' .join(('%', '%d.jpg')) % (i, count2)))))
65     PhotoCsv.append(open('/' .join((path, '/' .join(('%', '%d.csv')) % (i, count2)))))
66     Test[count1].append(PhotoCsv)
67
68     Test[count1].append(Name[count1])
69     count1=count1+1
70 return(Test)
71
72 #-----
73 def Taille(ImageColor):
74     r=90
75     g=140
76     b=80
77
78     ky=130
79     YHaut=[]
80     YBas=[]
81     for m in range(0,13,5):
82         yh=0
83         yb=0
84         for j in range(0,240):
85             if ImageColor.getpixel((ky+m,j))[0]>r and ImageColor.getpixel((ky+m,j))[1]<g and ImageColor.getpixel((ky+m,j))[2]<b:
86                 if yh==0:
87                     yh=j
88             YHaut.append(yh)
89             h=240
90             for i in range(0,240):
91                 h=h-1
92                 if ImageColor.getpixel((ky+m,h))[0]>r and ImageColor.getpixel((ky+m,h))[1]<g and ImageColor.getpixel((ky+m,h))[2]<b:
93                     if yb==0:
94                         yb=h
95             YBas.append(yb)
96
97     yh=YHaut[0]
98     yb=YBas[0]
99     for i in range(1,len(YHaut)):
100         if yh>YHaut[i]:
101             yh=YHaut[i]
102         if yb<YBas[i]:
103             yb=YBas[i]
104
105     if (yb-yh)%2==1:
106         yb=yb+1
107
```

ANNEXE I

```
108     kx=90
109     Xdroite=[]
110     Xgauche=[]
111     for m in range(0,13,5):
112         xd=0
113         xg=0
114         for j in range(0,320):
115             if ImageColor.getpixel((j,kx+m))[0]>r and ImageColor.getpixel((j,kx+m))[1]<g and ImageColor.getpixel((j,kx+m))[2]<b:
116                 if xg==0:
117                     xg=j
118             Xgauche.append(xg)
119             h=290
120             for i in range(0,290):
121                 h=h-1
122                 if ImageColor.getpixel((h,kx+m))[0]>r and ImageColor.getpixel((h,kx+m))[1]<g and ImageColor.getpixel((h,kx+m))[2]<b:
123                     if xd==0:
124                         xd=h
125             Xdroite.append(xd)
126
127     xd=Xgauche[0]
128     xg=Xdroite[0]
129     for i in range(1,len(Xdroite)):
130         if xd>Xgauche[i]:
131             xd=Xgauche[i]
132         if xg<Xdroite[i]:
133             xg=Xdroite[i]
134
135     if (xd-xg)%2==1:
136         xd=xd+1
137
138     centre=160-(xg+(xd-xg)//2)
139     return(yh,yb,centre)
140
141 #-----
142 def crop(Image):
143     yh,yb,decalage=Taille(Image)
144     box=(160-decalage-(yb-yh)//2,yh,160-decalage+(yb-yh)//2,yb)
145     ImageCropped=Image.crop(box)
146     return(ImageCropped)
147
148 #-----
149 def Fourier(Tableau):
150     T=[]
151     a=0
152     for row in Tableau:
153         a=a+1
154         F=[]
155         if len(row[0].split(';'))==1:
156             PartieEntiereRelou=row[0]
157         else:
158             for k in range(0,len(row[0].split(';'))-1):
159                 F.append(float(row[0].split(';')[k]))
160             PartieFractionnaireRelou=(row[1].split(';'))[0]
161             F.append(float('.'.join((PartieEntiereRelou,PartieFractionnaireRelou))))
162
```

ANNEXE I

```
163 k=0
164 while len(F)<320-len(row[-1].split(';')):
165     k=k+1
166     PartieEntiereEntier=[]
167     if len(row[k].split(';'))==2:
168         PartieEntiere=(row[k].split(';'))[1]
169     else:
170         PartieEntiere=(row[k].split(';'))[-1]
171         entier=0
172         while len(row[k].split(';'))-entier>2:
173             entier=entier+1
174             PartieEntiereEntier.append((row[k].split(';'))[entier])
175
176     PartieFractionnaire=(row[k+1].split(';'))[0]
177
178     if len(PartieEntiereEntier)!=0:
179         for i in range(0,len(PartieEntiereEntier)):
180             F.append(float(PartieEntiereEntier[i]))
181
182     if float('.'.join((PartieEntiere,PartieFractionnaire)))<10 or float('.'.join((PartieEntiere,PartieFractionnaire)))>40:
183         print(float('.'.join((PartieEntiere,PartieFractionnaire))))
184
185     F.append(float('.'.join((PartieEntiere,PartieFractionnaire))))
186
187     PartieEntiereRelou2=(row[-2].split(';'))[-1]
188     if len(row[-1].split(';'))!=1:
189         F.append(float('.'.join((PartieEntiereRelou2,row[-1].split(';'))[0])))
190         for i in range(1,len(row[-1].split(';'))):
191             F.append(float(row[-1].split(';')[i]))
192     else:
193         PartieFractionnaireRelou2=row[-1]
194         F.append(float('.'.join((PartieEntiereRelou2,PartieFractionnaireRelou2))))
195         T.append(F)
196
197     return(T)
198 #-----
199 def CSVcrop(T,Image):
200     Tcrop=[]
201
202     yh,yb,decalage=Taille(Image)
203     box=(160-decalage-(yb-yh)//2,yh,160-decalage+(yb-yh)//2,yb)
204
205     for i in range(box[1],box[3]):
206         TcropTempo=[]
207         for k in range(box[0],box[2]):
208             TcropTempo.append(T[i][k])
209         Tcrop.append(TcropTempo)
210     return(Tcrop)
211
212
```

ANNEXE I

```
213 #-----
214 def Moyenne(Matrice):
215     somme=0
216     for i in range(0,len(Matrice)):
217         for k in range(0,len(Matrice[i])):
218             somme=somme+Matrice[i][k]
219
220     moy=somme/(len(Matrice)*len(Matrice[0]))
221
222     for i in range(0,len(Matrice)):
223         for k in range(0,len(Matrice[i])):
224             Matrice[i][k]=Matrice[i][k]-moy
225
226 #-----
227 def MoyenneFFT(Liste):
228     SommeFFT=[]
229     for k in range(0,len(Liste[0])):
230         SommeFFT.append([0])
231         for i in range(0,len(Liste[0][0])-1):
232             SommeFFT[k].append(0)
233
234     for k in range(0,len(Liste[0])):
235         for i in range(0,len(Liste[0][0])):
236             for j in range(0,len(Liste)):
237                 SommeFFT[k][i]=SommeFFT[k][i]+Liste[j][k][i]
238
239     for k in range(0,len(SommeFFT)):
240         for i in range(0,len(SommeFFT[0])):
241             SommeFFT[k][i]=SommeFFT[k][i]/len(Liste)
242     return(SommeFFT)
243
244 #-----
245 def Distance(Liste):
246     ListeMin=[]
247
248     for k in range(0,len(Liste[0])):
249         ListeMin.append([0])
250         for i in range(0,len(Liste[0][0])-1):
251             ListeMin[k].append(0)
252
253     for k in range(0,len(Liste[0])):
254         for i in range(0,len(Liste[0][0])):
255             min=Liste[0][k][i]
256             for j in range(0,len(Liste)):
257                 if Liste[j][k][i]<min:
258                     min=Liste[j][k][i]
259             ListeMin[k][i]=min
260
261     ListeMax=[]
262
```

ANNEXE I

```
263     for k in range(0,len(Liste[0])):
264         ListeMax.append([0])
265         for i in range(0,len(Liste[0][0])-1):
266             ListeMax[k].append(0)
267
268     for k in range(0,len(Liste[0])):
269         for i in range(0,len(Liste[0][0])):
270             max=Liste[0][0][0]
271             for j in range(0,len(Liste)):
272                 if Liste[j][k][i]>max:
273                     max=Liste[j][k][i]
274             ListeMax[k][i]=max
275
276     return(ListeMin,ListeMax)
277
278 #-----
279 def ReduceFFT1(FFT):
280     FFT=np.fft.fft2(FFT)
281
282     FFTreduce=[]
283
284     FFT1=FFT[0:5]
285     for k in range(0,len(FFT1)):
286         FFTreduce.append((FFT[k])[5:])
287
288     FFT2=FFT[0:5]
289     for k in range(0,len(FFT2)):
290         FFTreduce[k]=np.concatenate((FFTreduce[k],FFT[k][len(FFT)-5:len(FFT)]))
291
292     FFT3=FFT[len(FFT)-5:]
293     for k in range(0,len(FFT3)):
294         FFTreduce.append((FFT[k])[5:])
295
296     FFT4=FFT[len(FFT)-5:]
297     for k in range(5,5+len(FFT4)):
298         FFTreduce[k]=np.concatenate((FFTreduce[k],FFT[k][len(FFT)-5:len(FFT)]))
299
300     FFT=np.array(FFTreduce)
301     FFT=abs(FFT)
302     return(FFT)
303
304 #-----
305 def ReduceFFT2(FFT):
306     FFT=np.fft.fft2(FFT)
307     FFT=np.array(FFT)
308     FFT=abs(FFT)
309     return(FFT)
310
```

ANNEXE I

```
311 #-----
312 def homogenisation(Liste):
313     taille=len(Liste[0])
314
315     ListeHomo=[]
316
317     for k in range(1,len(Liste)):
318         print('oui',len(Liste[k]))
319         if taille>len(Liste[k]):
320             taille=len(Liste[k])
321
322     for k in range(0,len(Liste)-1):
323         ListeHomo.append(np.zeros([taille,taille]))
324
325     for k in range(0,len(Liste)-1):
326         print('1',len(Liste[k]),len(Liste[k][0]))
327
328         if (len(Liste[k])-taille)%2==0:
329             x=(len(Liste[k])-taille)//2
330         else:
331             x=(len(Liste[k][i])-taille+1)//2
332
333         for i in range(0,taille):
334             if (len(Liste[k])-taille)%2==0:
335                 ListeHomo[k][i]=(Liste[k][i+x])[x:len(Liste[k][i])-x]
336             else:
337                 ListeHomo[k][i]=(Liste[k][i+x])[x:len(Liste[k][i])-x+1]
338
339         print('2',len(ListeHomo[k]),len(ListeHomo[k][0]))
340
341     return(ListeHomo)
342
343 #-----
344 def Comparaison1(FFT1,FFT2):
345     confiance1=0.3
346     taux1=0
347     for k in range(0,min(len(FFT1),len(FFT2))):
348         for i in range(0,min(len(FFT1[k]),len(FFT2[k]))):
349             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)):
350                 taux1=taux1+1
351
352     taux1=(taux1*100)/(min(len(FFT1),len(FFT2))*min(len(FFT1[0]),len(FFT2[0])))
353     return(taux1)
354
355 #-----
356 def Comparaison2(FFT1,FFT2):
357     confiance2=100
358     taux2=0
359     for k in range(0,min(len(FFT1),len(FFT2))):
360         for i in range(0,min(len(FFT1[k]),len(FFT2[k]))):
361             if FFT1[i][k]-confiance2<FFT2[i][k]<FFT1[i][k]+confiance2 or FFT2[i][k]-confiance2<FFT1[i][k]<FFT2[i][k]+confiance2:
362                 taux2=taux2+1
363
364     taux2=(taux2*100)/(min(len(FFT1),len(FFT2))*min(len(FFT1[0]),len(FFT2[0])))
365     return(taux2)
366
```

ANNEXE I

```
367 #-----
368 def ComparaisonDistance1(Liste,Intervalle):
369     distance=0
370
371     x=abs(len(Liste)-len(Intervalle[0]))/2
372
373     for k in range(0,min(len(Liste),len(Intervalle[0]))):
374         for i in range(0,min(len(Liste),len(Intervalle[0]))):
375             if Intervalle[0][k][i]<=Liste[k][i]<=Intervalle[1][k][i]:
376                 distance=distance+0
377             else:
378                 distance=distance+min(abs(Intervalle[0][k][i]-Liste[k][i]),abs(Intervalle[1][k][i]-Liste[k][i]))
379
380     return(distance/(min(len(Liste),len(Intervalle[0]))*min(len(Liste),len(Intervalle[0]))))
381
382 #-----
383 def ComparaisonDistance2(Liste,Intervalle):
384     distance=0
385
386     for k in range(0,min(len(Liste),len(Intervalle[0]))):
387         for i in range(0,min(len(Liste[0]),len(Intervalle[0][0]))):
388             distance=distance+max(abs(Intervalle[0][k][i]-Liste[k][i]),abs(Intervalle[1][k][i]-Liste[k][i]))
389
390     return(distance/(min(len(Liste),len(Intervalle[0]))*min(len(Liste),len(Intervalle[0]))))
391
392 #-----
393 def Axes(FFT):
394     X=[]
395     Y=[]
396     XX=[]
397     YY=[]
398
399     count=0
400     for row in FFT:
401         count=count+1
402         for k in range(0,len(row)):
403             X.append(count)
404             Y.append(k)
405             XX=np.concatenate((XX,X))
406             YY=np.concatenate((YY,Y))
407
408     X=[]
409     Y=[]
410     return(XX,YY)
411
412 #-----
413 def graph(courbe):
414     ax = plt.axes(projection='3d')
415
416     x,y=Axes(courbe)
417     ax.scatter(x, y, courbe, c='blue')
418
419     plt.xlabel('x - axe des abscisses ')
420     plt.ylabel('y - axe des ordonnées')
421     plt.grid()
422     ax.set_title('FFT')
423     plt.show()
424
425
```

ANNEXE I

```
426 #-----
427 def CSVManuel(Image):
428
429     Tmax=33.3
430     Tmin=13.0
431
432     Image=Image.crop((15,25,Image.size[0],Image.size[1]-27))
433
434     GradT=(Tmax-Tmin)/180
435
436     T=[]
437
438     startx=0
439     endx=Image.size[0]
440     starty=0
441     endy=Image.size[1]
442
443     echelle=[]
444     for y in range(5,183):
445         echelleK=[0,0,0]
446         for x in range(292,298):
447             for k in range(0,3):
448                 echelleK[k]=echelleK[k]+Image.getpixel((x,y))[k]
449             for k in range(0,3):
450                 echelleK[k]=int(echelleK[k]/5)
451             echelle.append(tuple(echelleK))
452
453     Croix=[]
454     for k in range(130,140):
455         for i in range(94,97):
456             Croix.append((k,i))
457
458     for k in range(151,160):
459         for i in range(94,97):
460             Croix.append((k,i))
461
462     for k in range(144,147):
463         for i in range(101,110):
464             Croix.append((k,i))
465
466     for k in range(144,147):
467         for i in range(80,90):
468             Croix.append((k,i))
469
470     for k in range(142,149):
471         for i in range(90,92):
472             Croix.append((k,i))
473
474     for k in range(142,149):
475         for i in range(99,101):
476             Croix.append((k,i))
477
478     for k in range(149,151):
479         for i in range(92,99):
480             Croix.append((k,i))
481
482     for k in range(140,142):
483         for i in range(92,99):
484             Croix.append((k,i))
485
```


ANNEXE I

```
486 Croix.append((141,91))
487 Croix.append((149,91))
488 Croix.append((142,92))
489 Croix.append((148,92))
490 Croix.append((142,98))
491 Croix.append((148,98))
492 Croix.append((141,99))
493 Croix.append((149,99))
494 for y in range(starty,endy):
495     Tx=[]
496     for x in range(startx,endx):
497         if (x,y) not in Croix:
498             value=True
499             value2=True
500             f=0
501             while value2:
502                 f=f+10
503                 for i in range(0,178):
504                     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:
505                         Tx.append(Tmin+(180-i)*GradT)
506                         Image.putpixel((x,y),echelle[i])
507                         value=False
508                         value2=False
509             else:
510                 Tx.append(Tx[-1])
511                 Image.putpixel((x,y),Image.getpixel((x-1,y)))
512
513     T.append(Tx)
514     return(T)
515
516 #-----
517 def CréationListe(Image,Fichier):
518     Tableau=csv.reader(Fichier)
519     ImageCropped=crop(Image)
520     CSV=Fourier(Tableau)
521     T=CSVcrop(CSV,Image)
522     Moyenne(T)
523     return(T)
524
525 #-----
526 def BaseDeDonnée1(Photo):
527     BDD=[]
528     for k in range(0,len(Photo)):
529         MoyenneK=[]
530         for i in range(0,len(Photo[k])-1):
531             MoyenneK.append(ReduceFFT1(CréationListe(Photo[k][i][0],Photo[k][i][1])))
532         MoyenneK=MoyenneFFT(MoyenneK)
533         BDD.append([Photo[k][-1],MoyenneK])
534
535     return(BDD)
536
```

ANNEXE I

```
537 #-----
538 def BaseDeDonnée2(Photo,PhotoTest):
539     BDD=[]
540     for k in range(0,len(Photo)):
541         ListeDistance=[]
542         for i in range(0,len(Photo[k])-1):
543             ListeDistance.append(CréationListe(Photo[k][i][0],Photo[k][i][1]))
544         for i in range(0,len(ListeDistance)):
545             ListeDistance[i]=ReduceFFT1(ListeDistance[i])
546         ListeDistanceBDD=Distance(ListeDistance)
547         BDD.append([Photo[k][-1],ListeDistanceBDD])
548
549     return(BDD)
550
551 ##
552 PhotoTest=Test()
553 Photo=Data()
554
555 BDD=BaseDeDonnée1(Photo)
556
557 count=0
558 for i in range(0,len(PhotoTest)):
559     TAUX=[]
560     FFTtest=ReduceFFT1(CréationListe(PhotoTest[i][0][0],PhotoTest[i][0][1]))
561     for k in range(0,len(BDD)):
562         TAUX.append([Comparaison1(BDD[k][1],FFTtest),BDD[k][0]])
563     ressemblance=0
564     gagnant=0
565
566     for k in range(0,len(TAUX)):
567         if TAUX[k][0]>=ressemblance:
568             ressemblance=TAUX[k][0]
569             gagnant=k
570
571     difference=ressemblance
572     for k in range(0,len(TAUX)):
573         if k!=gagnant:
574             if ressemblance-TAUX[k][0]<difference:
575                 difference=ressemblance-TAUX[k][0]
576                 posdiff=k
577
578     if TAUX[gagnant][1]==PhotoTest[i][1]:
579         count=count+1
580
581
582
```

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

```
641 BDD=BaseDeDonnée2(Photo,PhotoTest)
642
643 count=0
644 for i in range(0,len(PhotoTest)):
645     TAUX=[]
646     FFTtest=ReduceFFT1(CréationListe(PhotoTest[i][0][0],PhotoTest[i][0][1]))
647     for k in range(0,len(BDD)):
648         TAUX.append([ComparaisonDistance2(FFTtest,BDD[k][1]),BDD[k][0]])
649     distance=TAUX[i][0]
650     gagnant=0
651
652     for k in range(0,len(TAUX)):
653         if TAUX[k][0]<=distance:
654             distance=TAUX[k][0]
655             gagnant=k
656
657     posdiff=0
658
659     difference=TAUX[k][0]
660     for k in range(0,len(TAUX)):
661         if k!=gagnant:
662             if TAUX[k][0]-distance<difference:
663                 difference=TAUX[k][0]-distance
664                 posdiff=k
665
666     if TAUX[gagnant][1]==PhotoTest[i][1]:
667         count=count+1
668
669     TOTAL=TOTAL+count
670
671 print('Le nombre de bonne reconnaissance est',TOTAL,"pour 75 tests")
672 print("Soit un taux de",TOTAL*100/75,"%")
673
674
675 ##
676 TOTAL=0
677 for lol in range(0,5):
678     DATA=[]
679     Name=[]
680
681     for filename in os.listdir("C:/Users/Siena PAGE/Desktop/TIPE_BDD"):
682         Name.append(filename)
683
684     for k in range(0,len(Name)):
685         DATA.append([])
686
687     count1=0
688     for i in Name:
689         path="C:/Users/Siena PAGE/Desktop/TIPE_BDD"
690         path=('/' + i).join(("C:/Users/Siena PAGE/Desktop/TIPE_BDD", '%s')) % i)
691
692         count2=0
693         for filename in os.listdir(path):
694             type=filename.split('.')
695             if type[1]=='jpg':
696                 count2=count2+1
697
```

ANNEXE I

```
698     for k in range(1,count2+1):
699         if k!=lol+1:
700             PhotoCsv=[]
701             PhotoCsv.append(Image.open('/'.join((path, ''.join(('s', '%d.jpg')) % (i,k))))))
702             PhotoCsv.append(open('/'.join((path, ''.join(('s', '%d.csv')) % (i,k))))))
703             DATA[count1].append(PhotoCsv)
704
705         DATA[count1].append(Name[count1])
706         count1=count1+1
707
708     Test=[]
709
710     for k in range(0,len(Name)):
711         Test.append([])
712
713     count1=0
714     for i in Name:
715         path="C:/Users/Siena PAGE/Desktop/TIPE_BDD"
716         path=('/'.join(("C:/Users/Siena PAGE/Desktop/TIPE_BDD", '%s')) % i)
717         count2=lol+1
718         PhotoCsv=[]
719         PhotoCsv.append(Image.open('/'.join((path, ''.join(('s', '%d.jpg')) % (i,count2))))))
720         PhotoCsv.append(open('/'.join((path, ''.join(('s', '%d.csv')) % (i,count2))))))
721         Test[count1].append(PhotoCsv)
722
723         Test[count1].append(Name[count1])
724         count1=count1+1
725
726     Photo=DATA
727     PhotoTest=Test
728     BDD=BaseDeDonnée1(Photo)
729
730     count=0
731     for i in range(0,len(PhotoTest)):
732         TAUX=[]
733         FFTtest=ReduceFFT1(CréationListe(PhotoTest[i][0][0],PhotoTest[i][0][1]))
734         for k in range(0,len(BDD)):
735             TAUX.append([Comparaison1(BDD[k][1],FFTtest),BDD[k][0]])
736             ressemblance=0
737             gagnant=0
738
739             for k in range(0,len(TAUX)):
740                 if TAUX[k][0]>=ressemblance:
741                     ressemblance=TAUX[k][0]
742                     gagnant=k
743
744             difference=ressemblance
745             for k in range(0,len(TAUX)):
746                 if k!=gagnant:
747                     if ressemblance-TAUX[k][0]<difference:
748                         difference=ressemblance-TAUX[k][0]
749                         posdiff=k
750
751             if TAUX[gagnant][1]==PhotoTest[i][1]:
752                 count=count+1
753             TOTAL=TOTAL+count
754
755     print('Le nombre de bonne reconnaissance est',TOTAL,"pour 75 tests")
756     print("Soit un taux de",TOTAL*100/75,"%")
```

ANNEXE II

```
1 import cv2
2 import os
3 from PIL import Image
4 import numpy as np
5 import PIL
6 import time
7
8 def croix():
9     Croix=[]
10    for k in range(130,140):
11        for i in range(94,97):
12            Croix.append((k,i))
13
14    for k in range(151,160):
15        for i in range(94,97):
16            Croix.append((k,i))
17
18    for k in range(144,147):
19        for i in range(101,110):
20            Croix.append((k,i))
21
22    for k in range(144,147):
23        for i in range(80,90):
24            Croix.append((k,i))
25
26    for k in range(142,149):
27        for i in range(90,92):
28            Croix.append((k,i))
29
30    for k in range(142,149):
31        for i in range(99,101):
32            Croix.append((k,i))
33
34    for k in range(149,151):
35        for i in range(92,99):
36            Croix.append((k,i))
37
38    for k in range(140,142):
39        for i in range(92,99):
40            Croix.append((k,i))
41
42    Croix.append((141,91))
43    Croix.append((149,91))
44    Croix.append((142,92))
45    Croix.append((148,92))
46    Croix.append((142,98))
47    Croix.append((148,98))
48    Croix.append((141,99))
49    Croix.append((149,99))
50
51    return(Croix)
52
```

ANNEXE II

```
55 video='Desktop/TipeNelloVideo.mp4'
56 path_output_dir='Desktop/VideoTestTipe'
57
58 vidcap = cv2.VideoCapture(video)
59
60 fps=vidcap.get(cv2.CAP_PROP_FPS)
61
62 Croix=croix()
63
64 start_time = time.time()
65
66 nbImageSeconde=2
67
68 ListeImage=[]
69
70
71 count = 0
72 while vidcap.isOpened():
73     success, image = vidcap.read()
74     if success:
75         cv2.imwrite(os.path.join(path_output_dir, '%d.png') % count, image)
76         count=count+1
77         if count%(30//nbImageSeconde)==0:
78             ImageTest=Image.open("Desktop/videoTestTipe/%d.png" % count)
79             ListeImage.append(ImageTest.crop((15,25,ImageTest.size[0]-20,ImageTest.size[1]-27)))
80
81     else:
82         break
83 cv2.destroyAllWindows()
84 vidcap.release()
85
86
87
88 for k in range(0,4):
89     ListeImage.pop()
90
91 rayon=100
92
93 lastpos=(0,0)
94 ListePos=[]
95
```

ANNEXE II

```
96 for k in range(0,len(ListeImage)):
97     pos=True
98     image=ListeImage[k]
99
100     if lastpos[0]-rayon//2>=0:
101         xmin=lastpos[0]-rayon//2
102     else:
103         xmin=0
104
105     if lastpos[0]+rayon//2<=image.size[0]:
106         xmax=lastpos[0]+rayon//2
107     else:
108         xmax=image.size[0]
109
110
111     if lastpos[1]-rayon//2>=0:
112         ymin=lastpos[1]-rayon//2
113     else:
114         ymin=0
115
116     if lastpos[1]+rayon//2<=image.size[1]:
117         ymax=lastpos[1]+rayon//2
118     else:
119         ymax=image.size[1]
120
121
122     if k==0:
123         xmin=0
124         ymin=0
125         xmax=image.size[0]
126         ymax=image.size[1]
127
128
129     for x in range(xmin,xmax):
130         for y in range(ymin,ymax):
131             if (x,y) not in Croix:
132                 if 200<image.getpixel((x,y))[0]<260 and 40<image.getpixel((x,y))[1]<260 and 100<image.getpixel((x,y))[2]<260:
133                     if pos==True:
134                         pos=(x,y)
135                         lastpos=pos
136                         print((time.time() - start_time))
137             ListePos.append(pos)
138
139
140 for k in range(0,len(ListePos)):
141     for j in range(0,5):
142         for i in range(0,5):
143             if ListePos[k][0]+j>=ListeImage[k].size[0] or ListePos[k][1]+i>=ListeImage[k].size[1]:
144                 test=0
145             else:
146                 ListeImage[k].putpixel((ListePos[k][0]+j,ListePos[k][1]+i),(0,0,0))
147
148 for i in range(0,len(ListeImage)):
149     ListeImage[i].save("Desktop/videoTipeSave/%d.jpg" % i)
150
```